高等学校试用教材

True BASIC 语言
模块化结构化
程序设计

邓德祥

高等教育出版社

True BASIC 语言模块化结构化程序设计 邓德祥

高等学校试用教材

True BASIC 语言
模块化结构化程序设计

邓 德 祥

TP31
77

高等教育出版社

内 容 提 要

本书全面地介绍了符合美国国家标准的开拓 BASIC 新纪元的 True BASIC 语言，同时介绍了结构化程序设计的基本方法和结构化流程图，以及在 IBM 微型机上True BASIC的上机操作方法.

本书以结构化程序设计为主线，针对初学者的特点对各部分内容作了较合理的安排. 本书的特点是由浅入深、循序渐进、例题丰富、概念清楚、言简意明. 学习本书可满足各种科学计算、数据处理和画图的需求.

本书可作为高等学校计算机基础教育的教材，也可供中专、职工大学、职业高中和短训班使用，还可供各类工程技术人员、管理干部自学或参考.

JS464/27

高等学校试用教材

True BASIC 语言

模块化结构化程序设计

邓德祥

*

高等教育出版社出版

新华书店北京发行所发行

河北省香河县印刷厂印装

*

开本 787×1092 1/16　　印张 15.25　字数 350 000

1988 年 10月第 1 版　1988 年 10月第 1 次印刷

印数 0 001—[illegible]110

ISBN7-04-001634-6/TP·25

定价 3.60 元

前　言

BASIC 语言是美国计算机语言学家 John G. Kemeny 和 Thomas E. Kurtz 吸收FORTRAN Ⅳ 和 ALGOL60 语言的优点于 1964 年推出的. 由于它具有小型、通用、会话、易学的特点,因而很快得到了推广和应用. 但由于它是非标准化的语言, 因而 BASIC 程序通用性较差. 另外,近年来结构化程序设计方法已成为受软件工作者欢迎的程序设计方法, 但原 BASIC 语言功能差,不能适应结构化程序设计方法的要求.

1984 年美国公布了"美国国家标准 BASIC 程序设计语言". Kemeny 和 Kurtz 等于 1985 年推出了严格按照美国国家标准实现的模块化、结构化程序设计语言——True BASIC.

True BASIC(真的 BASIC)保持了原 BASIC 语言"小型、通用、会话、易学"的特点. 和当代其它的高级语言比较, 它具有语言功能强、语法规定自然、程序结构清晰的特点, 而且具有良好的操作环境. 因此 True BASIC 被誉为"开拓 BASIC 新纪元的软件". 学会了 True BASIC 完全可以满足各种科学计算、数据处理和画图的需要. 为使广大科技工作者、管理干部和青年能尽快地掌握它,作者对 True BASIC 软件进行了消化和开发,编写调试了各种程序一百多个, 基于多年教学和编写教材的经验,按照高等学校教学的要求,写成了本书奉献给读者.

本书包含了 True BASIC 语言、结构化程序设计方法和上机操作方法三部分内容. 结构化程序设计能力的培养是本书的出发点和归宿. 本书以结构化程序设计为主线, 根据程序设计的要求逐步引入了各种语句. 全书共分 12 章. 第 1, 2 章讲述了学习本书的基本知识. 第 3 章到第 5 章分别讲述了顺序、选择和循环结构程序设计. 第 6 章讲述了数组与矩阵运算. 第 7 章讲述了画图功能. 第 8 章讲述了函数、子程序和图画三种模块和库的设计. 第 9 章讲述了正文文件、字节文件和记录文件的使用. 第 10 章讲述了音响程序设计. 第十一章讲述了其它程序设计,供进一步提高时阅读. 第 12 章讲述了上机操作方法,包括IBM 微机系统介绍, 操作系统基本命令,True BASIC 程序的输入、调试和运行,以及汉字的输入和输出.

本书内容齐全,例题丰富,由浅入深,循序渐进,可作为高等学校计算机基础教育的教材, 也可供职工大学和有关短训班使用,还可供各类工程技术人员自学或参考.

全国高等学校计算机基础教育研究会副会长谭浩强教授担任了本书的主审工作, 北京自动化学院田淑清副教授阅读了书稿, 北方工业大学田成方教授阅读了初稿,他们都给了很大支持,并提出了许多宝贵意见,在此表示诚挚的感谢.

由于作者水平有限、资料甚少、时间仓促, 错漏和不妥之处在所难免, 敬请各位读者批评指正.

作者

于1988.2

目　录

第1章　计算机、语言和程序设计……………1

1.1　计算机简介……………1

1.1.1　计算机的特点……………1

1.1.2　计算机的组成……………2

1.1.3　计算机运行的示意过程……………2

1.2　计算机语言概述……………3

1.2.1　机器语言……………3

1.2.2　汇编语言……………3

1.2.3　高级语言……………4

1.2.4　超高级语言……………5

1.3　程序设计概述……………5

1.3.1　明确问题……………5

1.3.2　建立模型……………5

1.3.3　算法设计……………5

1.3.4　编写程序……………6

1.3.5　编译运行……………6

1.3.6　结果分析……………6

1.4　程序设计的辅助手段——控制流程图……………7

1.5　结构化程序设计简介……………8

1.5.1　三种基本结构……………8

1.5.2　结构化程序设计准则……………9

1.5.3　结构化程序设计方法……………9

1.5.4　结构化流程图——N-S图……………9

1.5.5　评价程序的基本准则……………12

1.6　True BASIC软件的优点……………12

1.6.1　模块化结构化多功能的程序设计语言……………12

1.6.2　简便宽松的操作环境……………13

习　题……………13

第2章　True BASIC语言初步知识……………15

2.1　字符集与符号名……………15

2.1.1　字符集……………15

2.1.2　符号名……………16

2.2　数据……………17

2.2.1　数值常数……………17

2.2.2　字符串常数……………18

2.2.3　简单变量和赋值……………18

2.2.4　变量赋值语句……………19

2.3　语句……………19

2.3.1　简单语句与结构语句……………19

2.3.2　可执行语句与非执行语句……………20

2.4　源程序……………20

2.4.1　源程序的结构……………20

2.4.2　源程序的书写格式……………20

习　题……………21

第3章　顺序程序设计……………22

3.1　表达式……………22

3.1.1　算术表达式……………22

3.1.2　字符串表达式……………24

3.2　PRINT语句……………26

3.2.1　数值和字符串的显示……………27

3.2.2　逗号区域定位法……………27

3.2.3　tab定位法……………28

3.3　PRINT USING语句……………29

3.3.1　PRINT USING和数值……………30

3.3.2　PRINT USING和字符串……………33

3.4　打印输出……………33

3.5　INPUT语句……………34

3.5.1　常用的INPUT语句格式……………34

3.5.2　LINE INPUT语句……………35

3.6　顺序程序应用举例……………36

习　题……………38

第4章　选择程序设计……………40

4.1　逻辑表达式……………40

4.1.1　关系表达式……………

4.1.2　逻辑表达式……………

4.1.3　逻辑函数……………

4.2　IF语句……………

4.2.1　简单IF语句……………

4.2.2 结构IF语句……44
4.3 CASE语句……47
4.3.1 SELECT CASE语句(选择情况语句)的一般形式……47
4.3.2 应用举例……48
习 题……49
第5章 循环程序设计……51
5.1 循环概述……51
5.2 条件型循环……53
5.2.1 当型循环……53
5.2.2 直至型循环……54
5.2.3 带EXIT的无限循环……55
5.2.4 带EXIT的当型循环或直至型循环……55
5.2.5 条件型循环应用举例……56
5.3 计数型循环……60
5.3.1 计数型循环结构语句形式……60
5.3.2 计数型循环的几点说明……61
5.3.3 计数型循环应用举例……62
5.4 READ语句与DATA语句……65
5.4.1 READ和DATA语句一般形式……65
5.4.2 DATA数据项系统判断条件……66
5.4.3 DATA数据的重复使用……67
5.5 循环结构程序设计举例……68
习 题……73
第6章 数组及其应用……75
6.1 数组概述……75
6.1.1 数组变量的引入……75
6.1.2 数组说明……76
6.1.3 下标变量……76
6.2 数组赋值与运算……78
6.2.1 数组赋值……78
6.2.2 数组运算……78
6.2.3 系统数组……80
6.2.4 有关数组的函数……80
6.3 数组的输入/输出语句……81
6.3.1 MAT PRINT语句……81
6.3.2 MAT PRINT USING语句……82
6.3.3 MAT INPUT语句……82
6.3.4 MAT LINE INPUT语句……83
6.3.5 MAT READ语句……84
6.4 可调数组……85
6.4.1 显式调整……85
6.4.2 隐式调整……86
6.5 数组应用……88
习 题……94
第7章 图形……96
7.1 图形坐标……96
7.1.1 图形坐标设置语句……96
7.1.2 询问图形坐标语句……97
7.2 基本画图语句……97
7.2.1 画点……97
7.2.2 画线……98
7.2.3 画面……100
7.2.4 在图形坐标上字符的输出……102
7.3 图形颜色的设置与询问……103
7.3.1 前景颜色的设置与询问……103
7.3.2 后景颜色……103
7.4 BOX语句……104
7.4.1 画矩形框……104
7.4.2 画矩形面……105
7.4.3 画椭圆……106
7.4.4 FLOOD语句……107
7.4.5 动画……108
7.5 GET语句……111
7.5.1 GET KEY语句……111
7.5.2 GET POINT语句……112
7.5.3 GET MOUSE语句……114
7.6 字符坐标与光标的设置和询问……115
习 题……116
第8章 辅程序与库……118
8.1 辅程序概述……118
8.1.1 辅程序的种类……118
8.1.2 辅程序的特性……118
8.1.3 辅程序的执行……118
8.1.4 引入辅程序的优点……119
8.2 子程序的定义与调用……119
8.2.1 子程序定义的一般形式……119
8.2.2 子程序的调用……119
8.2.3 子程序应用举例……119
8.3 函数的定义与调用……121
8.3.1 函数定义的一般形式……121
8.3.2 函数的调用……121

8.3.3 函数应用举例……122
8.3.4 子程序与函数的比较……124
8.4 虚实结合……125
8.4.1 虚实结合概述……125
8.4.2 值结合与名结合……125
8.4.3 数组虚实结合……125
8.5 全局变量与局部变量……127
8.6 递归……132
8.6.1 直接递归……132
8.6.2 间接递归……137
8.7 图画的定义与调用……138
8.7.1 图画定义的一般形式……138
8.7.2 图画调用形式之一……139
8.7.3 图画调用形式之二……141
8.7.4 图画调用形式之三……143
8.8 打开、关闭和转移窗口……147
8.8.1 有关窗口语句……147
8.8.2 多窗口应用举例……148
8.9 库文件……151
8.9.1 库的形式及辅程序的调用……151
8.9.2 标准库……152
习 题……155
第9章 文件……157
9.1 文件概述……157
9.1.1 文件种类……157
9.1.2 通道与文件……157
9.1.3 文件指针的位置……157
9.1.4 文件数据的测试……158
9.1.5 文件存取模式……158
9.1.6 文件建立模式……159
9.1.7 文件的组织模式……159
9.1.8 文件的大小……159
9.2 文件的通用语句……159
9.2.1 OPEN 语句……160
9.2.2 CLOSE 语句……160
9.2.3 ERASE 语句……160
9.2.4 UNSAVE 语句……160
9.3 正文文件……161
9.3.1 正文文件的范围与域宽……161
9.3.2 正文文件的读写……161
9.3.3 正文文件应用举例……162
9.4 记录文件……163
9.4.1 记录长度的设置与询问……164
9.4.2 记录文件的读写……164
9.4.3 记录文件的应用举例……166
9.5 字节文件……171
9.5.1 字节文件的读写……171
9.5.2 应用举例……172
习 题……174
第10章 音响……175
10.1 PLAY 语句……175
10.2 SOUND 语句……179
第11章 其它程序设计……181
11.1 出错与出错处理……181
11.1.1 出错处理结构语句……181
11.1.2 出错处理函数……182
11.1.3 设置出错代码和信息语句……183
11.2 DO 程序……186
11.2.1 DO 程序的建立……186
11.2.2 DO 程序的编译和存盘……186
11.3 压缩辅程序……187
11.3.1 Packb 子程序……187
11.3.2 Unpackb 函数……188
11.3.3 压缩子程序的示例……188
11.4 链接程序……189
11.4.1 链接程序形式之一……189
11.4.2 链接程序形式之二……190
11.5 汇编语言程序的调用……190
第12章 上机操作方法……192
12.1 IBM PC 系统简介……192
12.1.1 硬件……192
12.1.2 软件……195
12.1.3 开机与关机……196
12.1.4 DOS系统常用命令……196
12.2 实验前的准备……198
12.2.1 复制 True BASIC 软盘……198
12.2.2 把 DOS 复制到新拷贝的 True BASIC 软盘上……198
12.2.3 True BASIC 系统主要程序……199
12.3 True BASIC 的进入与退出……
12.3.1 进入 True BASIC……
12.3.2 屏幕上的窗口……

12.3.3 退出 True BASIC…………………201
12.4 常用 True BASIC 文件处理命令和程序控制命令…………………201
12.5 初级编辑…………………202
12.5.1 程序(文件)的编辑…………………202
12.5.2 True BASIC 命令的编辑…………203
12.6 高级编辑…………………204
12.6.1 Find (查找)命令…………………204
12.6.2 Change(置换)命令…………………205
12.6.3 Try(试探)命令…………………205
12.6.4 行模块编辑…………………205
12.7 程序的调试…………………206
12.8 汉字输入与输出简介…………………208
12.8.1 汉字输入…………………208
12.8.2 汉字输出…………………209
附录 A 错误信息表…………………211
A.1 出错解释…………………211
A.2 出错代码及出错信息…………………221
附录 B ASCII 码符号集…………………224
附录 C True BASIC 语句一览表…………226

第1章 计算机、语言和程序设计

随着计算机的普及和推广，有愈来愈多的人迫切地希望学会使用计算机．人们步入“计算机世界”，自然要使用一种工具与计算机交流信息，这个工具通称为计算机语言．作科学计算和事务处理，通常使用过程型命令语言，人们不仅要告诉计算机“做什么”，还要告诉计算机“怎样做”．怎样做的步骤要用一系列的语句(或指令)表示，这个能完成一定任务的语句(或指令)序列就叫程序．而编写程序的过程就叫程序设计．计算机的一切操作是按程序执行的．所以，要想使用电子计算机，就要学会计算机语言程序设计．

BASIC 语言是国内外广泛应用的计算机语言之一．True BASIC 是 BASIC 创始人 John G. Kemeny 和 Thomas E. Kurtz 等人于 1985 年严格按照 1984 年“美国国家标准 BASIC 程序设计语言”实现的．它一方面吸收了现有计算机高级语言的优点，另一方面又保持了 BASIC 的小型、通用、会话、易学的特点．所以命名为真(true)BASIC．True BASIC 被誉为开拓 BASIC 语言新纪元的软件，它是模块化、结构化的程序设计语言．

这一章先简单地介绍一下计算机的组成，计算机语言和程序设计，为学习 True BASIC语言程序设计奠定基础．

1.1 计算机简介

世界上第一台电子计算机 ENIAC 是 1946 年问世的．计算机的发展非常迅速，应用特别广泛，对于人类社会的发展起着巨大的推动作用，致使人类社会进入了一个新阶段——“信息社会”．

1.1.1 计算机的特点

(1) 具有计算能力．计算速度快、精度高、可靠性强．

计算机的计算速度已经可以达到每秒几十亿次，一般为每秒几百万次到每秒几千万次．这样的速度是人手工计算不能比拟的．计算机的计算精度一般为有效数字 6 位，也可以为十几位，甚至上百位．这也是任何其它计算工具所望尘莫及的．计算机的可靠性很强，它可以连续工作几个月甚至几年不出错误．

(2) 具有记忆能力．计算机能存储程序和数据．

(3) 具有逻辑判断能力．

计算机能对两个信息进行比较，根据比较结果决定下一步的操作，实现自动控制．

概括地说，计算机是一种能高速、自动、准确进行“计算”(信息处理)的电子设备．已被广泛

地用于科学计算、数据处理(DP)、自动控制、人工智能(如机器人、专家系统)和计算机辅助设计(CAD),计算机辅助教学(CAI)等方面.

1.1.2 计算机的组成

计算机并不神秘,它的组成可分两大部分,一部分是机器系统(硬件),另一部分是程序系统(软件).

(1) 硬件.

如图1-1所示,其中←—表示数据传送路线,←……表示控制信号传送路线.计算机硬件分五大部分.

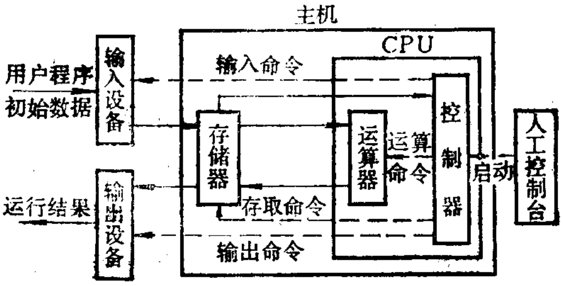


图1-1 计算机硬件结构示意图

a) 运算器.主要用来做算术运算和逻辑运算,相当于算盘.

b) 存储器.用来存储程序和数据,具有记忆功能,相当于纸.存储器的特点是"取之不尽,存之更新".

c) 控制器.通过程序的执行,控制、协调各部分的操作.

d) 输入设备.用来输入程序和初始数据等,常用设备有键盘、磁盘机、磁带机、卡片输入机和光电机(纸带)等.

e) 输出设备.用来输出程序清单、计算结果等.常用设备有显示器、打印机、磁盘机、磁带机、卡片输出机、穿孔机(纸带)和绘图仪等.

运算器和控制器合起来叫中央处理机(CPU).CPU加上存储器叫主机.输入和输出设备统称为外部设备.计算机的机器系统又称为裸机,或称为硬件.

一台裸机要正常工作和充分发挥效能,还需要配上功能完善的程序系统(软件),以使计算机能按照用户的指令自动地进行各种操作.

(2) 软件.

确切地说,软件是程序系统和文档资料(有关程序资料及使用说明等)的总称.这里是把文档资料看作程序不可缺少的附件,关于这一点今后不再说明.简单地说,软件是程序的总称,它可分为:

a) 系统程序(系统软件).常指与硬件配套出售的基本软件,包括管理整个计算机的操作系统,翻译各种语言的编译系统以及检查、诊断计算机自身的服务系统等.

b) 应用程序(应用软件).为便于使用计算机而开发的程序,通常以软件包的形式出现.例如,数学计算软件包、绘图软件包和企业管理软件包等.

有了硬件和软件就构成了计算机完整的系统.那么这样一台计算机又如何工作的呢?

1.1.3 计算机运行的示意过程

如图1-1所示,计算机运行过程可分为如下几个步骤:

a) 启动(开机).通过人工控制使机器进入系统软件控制之下.

b) 输入用户程序和初始数据.使用计算机的人用计算机语言表示的解题步骤称为用户

程序.

c) 运行用户程序,输出计算结果.

d) 如果还有程序运行则重复第二、第三步,直至所有程序运行完毕

e) 关机.

1.2 计算机语言概述

1.2.1 机器语言

如前所述,计算机语言是人与计算机交流信息的工具.这种工具最初级的形式就是二进制代码,即按特定规则由0和1组成的一定位数的数字代码.它能控制计算机进行特定的动作(例如加、减、取数、存数等).由于这种二进制代码能被计算机识别,所以称为机器指令.这些指令的集合称为机器语言.机器语言是因机而异的,就是不同机型,机器语言不同,因此可移植性差.用机器语言写出来的程序称为机器语言程序.这种程序的优点是计算机能直接运行,效率高.例如,求两个数值之和(56+43),用Z-80指令编写的程序如表1-1所示.

表1-1 计算56+43的Z-80机器语言程序

存储单元地址(16进制数①)	存储单元内容(二进制)	说明
00 40 00 41	00 11 11 10 00 11 10 00 (56)	取56送A累加器
00 42 00 43	00 00 01 10 00 10 10 11 (43)	取43送B寄存器
00 44	10 00 00 00	把B加到A上
00 45 00 46	00 11 00 10 01 00 10 00	把A内容送到0048单元
00 47 00 48	00 00 00 00 01 10 00 10 (99)	空操作(存结果的单元)
00 49	01 11 01 10	停机

① 16进制的基数 0 1 2 3 4 5 6 7 8 9 A B C D E F

对应的10进制数 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

作为一个用户,最关心的是如何学习和使用计算机语言.通过上述小小的例子可以看出,机器语言难学难用,而且易错难改,十分枯燥无味.为了便于记忆,计算机专家们又创造了汇编语言.

1.2.2 汇编语言

汇编语言是用特定的助记符号代表数字代码,帮助人们记忆,它和机器指令基本上是一一对应的.这个改进不算大,所以人们把机器语言和汇编语言通称为低级语言.上述计算用Z-80汇编语言编写的程序如表1-2所示,汇编程序语义和机器语言程序一一对应.

表 1-2　计算 56+43 的 Z-80 汇编程序

汇编语言程序	说　　明
START'LD A, 38H'	38H 表示38是16进制数
LD B, 2BH	2BH表示2B 是16进制数
ADD A,B	A 为累加器
LD (48H),A	B 为寄存器
NOP	空语句
HALT	停语句

1.2.3　高级语言

第一台电子计算机问世后，又经过十年的努力，到 1956 年，在 IBM704 机上实现了 FORTRAN1．这是由低级语言到高级语言的飞跃．高级语言之所以称之为“高级”是因为它是用英语和人們熟悉的数学公式来表达的，并且又能为计算机所接受，具有较好的通用性．上述例子的 True BASIC 程序，如表 1-3 所示．

表 1-3　计算 56+43 的 True BASIC 程序

True BASIC程序	说　明
PROGRAM ex01	程序开头语句
LET a=56	把56赋给 A
LET b=43	把43赋给 B
PRINT a+b	打印出A+B的值
END	结束

这里存在一个问题，即上面提到电子计算机能直接识别二进制代码，但又如何识别这些字符呢？事实上，高级语言程序，通常称为源程序，输入计算机之后，先要被翻译成机器语言程序，通常称为目的程序，然后再执行．这个翻译工作是由一个系统软件完成的．

True BASIC 有两种方式：一种叫解释程序，这类似于口头翻译，它是边解释边执行，直接用 RUN(或按[F9])命令运行．另一种叫编译程序，这类似于文件翻译，生成一个机器语言的目标程序，再用RUN(或按[F9])命令运行．

编译程序的功能，如图 1-2 所示．

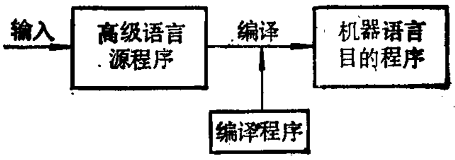


图 1-2　编译程序的功能

目前国内外常用的高级语言有十几种，分别具有不同特点，例如：

BASIC	(小型会话式语言)
True BASIC	(模块化结构化程序设计语言)
FORTRAN 66	(适用于科学计算)

FORTRAN 77	(适用于科学计算、数据处理)
COBOL	(适用于数据处理)
PL/1, ALGOL 68	(大型通用语言)
PASCAL	(结构化程序设计语言)
C	(系统程序设计语言)

1.2.4 超高级语言

高级语言经过 30 年的发展，目前可以说已经接近登峰造极的地步，Ada 语言的出现就是一个明显的标志. 高级语言是过程型命令语言，以赋值为核心，适用于数值/数据处理. 而超高级语言是非过程型命令语言，有逻辑型、函数型以及逻辑/函数混合型等. 以 PROLOG 语言为例，它是以模式匹配与回溯求解为核心，适用于知识信息处理. 它一改传统过程语言的风格，人们只需要把问题所涉及的事物之间的逻辑关系描述清楚，而不必给出求解问题的具体步骤，只要向系统提出问题，系统将自动地求出解答. 1981年日本宣布：他们研制的第五代计算机已将PROLOG语言选作核心语言.

1.3 程序设计概述

如前所述，学习计算机语言的目的是用它来编写程序，解决科学计算、事务处理等问题. 下面通过一个例子说明用计算机解题过程，例题中的一些细节，当前不必深究，以后自然明白.

1.3.1 明确问题

在着手解决问题之前，首先要弄清问题的条件和最终要求.

例如，"若 A,B 两点有着不可逾越的障碍，求 A,B 两点之间的距离". 这样的问题条件和要求都很明确. 如果问题是"求两点之间的距离"，要求虽明确，但条件不太明确. 若能直接用米尺测量，那么，一般无需编程序用计算机计算了.

1.3.2 建立模型

一个现实的问题，往往比较复杂，其影响因素是多方面的，这时要进行试验分析，抓住主要矛盾，丢掉次要矛盾，把原来的问题化作一个理想的模型. 例如，对于上述测量 A,B 两点间距离这样一个问题，根据测量学，选择一点 C，如图 1-3 所示. 于是可测得：

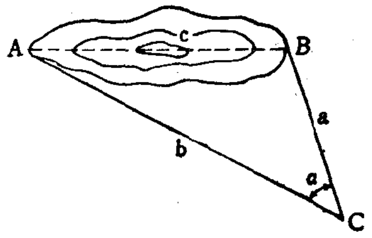


图 1-3

$$a=1.5,\ b=3.2,\ c=0.8\text{(弧度)}$$

这样就把实际问题化为"已知三角形两边和夹角，求第三边 c" 的数学问题了. 这一步也叫建立数学模型.

1.3.3 算法设计

所谓算法设计，就是把数学模型规定的任务，转化为具体的解题步骤，即产生解题的逻辑蓝

图．如果说数学模型规定“做什么”，那么算法设计就是解决“怎样做”．例如对于上述问题，由三角学知道，

$$c=\sqrt{a^2+b^2-2ab\cos\alpha}$$

把已知值代入，即可算出 c．

1.3.4 编写程序

编写程序就是用计算机语言把算法描述出来，如上述问题用 True BASIC 语言可编程序如下：

```
1 PROGRAM ex01-1
2 REM computing the side of a triangle
3       PRINT "a, b, alpha? "
4       INPUT a, b, alpha
5       LET d=a*a+b*b-2*a*b*cos(alpha)
6       LET c=sqr(d)
7       PRINT "a=";a, "b=";b,"alpha=";alpha,"c=";c
8 END
```

True BASIC 程序一句一行，一行也只能写一句．第 1 行是主程序开始语句；第 2 行是注释行；第 3 行是打印语句，输出一个字符串，作为第 4 行的提示；第 4 行是键盘输入语句；第 5，6 行是赋值语句；第 7 行打印输出结果；第 8 行结束语句．

1.3.5 编译运行

这一步也叫上机操作．第一步把编好的程序输入计算机内存；第二步编译运行：在系统软件支持下自动地先检查语法，后执行．如果发现“编译错误”（语句不符合语法规定），则编译中断，这时光标移至程序出错处，并给出“出错信息”．根据出错信息提示，进行修改，然后再运行，逐步去掉语法错误．上述程序运行时，

显示：a, b, alpha?

输入：1.5, 3.2, 0.8↙（↙表示 enter 键）

显示：a=1.5 b=3.2 alpha=0.8 c=2.408

如果输入数据不合理，还会出现“运行错误”（在程序运行过程中出现的错误），如溢出等．

1.3.6 结果分析

通过结果分析，可以检查是否还隐藏着逻辑性错误（程序未准确表示算法或题意，如用 0 除等）．上述解题过程，可以概括地用图 1-4 表示．

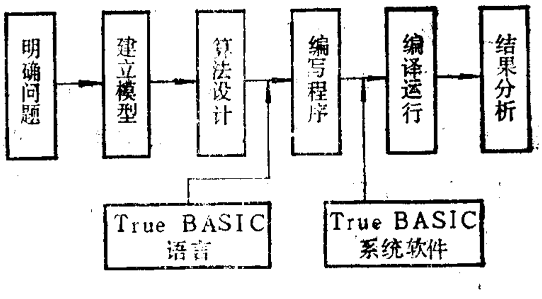


图 1-4 用计算机解题步骤

本课程的重点是讲述 True BASIC 语言和程序设计，其它步骤也会涉及到，但不作为重点．例如编译运行，将涉及到计算机原理，操作系统和编译方法等课程的知识，已超出本书范围，不宜详述．

1.4 程序设计的辅助手段——控制流程图

对一个稍微复杂的问题，根据算法直接写出程序是不太容易的．人们常用图形把解题步骤形象直观地描述出来，这样的图形就叫控制流程图．简单地说，流程图是算法的逻辑蓝图，是算法与程序之间的桥梁．画流程图是编写程序和阅读程序时常用的辅助手段．除此之处，还可采用 N-S 图，或伪代码等辅助手段．

例如，已知三角形的两边(a 和 b)和夹角(α)，求第三边(c)，初始数据如下:

a	b	α 弧度
1.2	2.0	0.2
2.0	4.0	0.5
0.0	0.0	0.0(结束标志)

可画流程图，如图 1-5 所示．

按图 1-5 可编程序如下:

```
PROGRAM ex01_2
REM computing the side of a triangle
DECLARE DEF side
PRINT '    a    b    alpha'
INPUT a,b,alpha
DO while a>0 and b>0 and alpha>0
     LET c=side (a,b,alpha)
     PRINT a,b,alpha,'c=';c
     INPUT a,b,alpha
LOOP
END
! function subprogram
DEF side (a1,b1,,alpha1)
     LET d=a1*a1+b1*b1-2*a1*b1*cos(alpha1)
     LET side=sqr(d)
END DEF
```

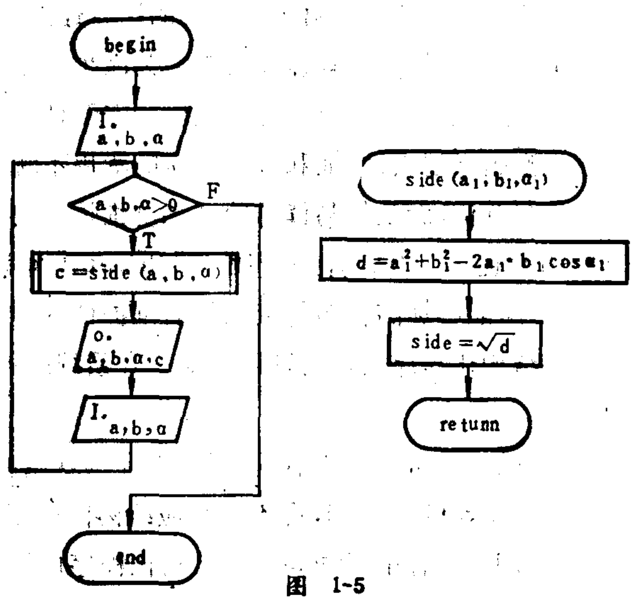


图 1-5

在流程图中，程序的逻辑关系和处理顺序是以一个框代表一段程序(包括一个或几个语句)，以箭头表示程序执行方向．在程序设计时，流程图可先粗后细，逐步细化，直到可方便地写出程序．在阅读别人的程序时，也可以把程序执行过程用流程图表示出来，便于阅读．总之借助流程图编写或阅读程序，能使程序逻辑关系清楚，层次分明，减少差错．图 1-6 是流程图中常用的符号，这与 1963 年美国标准协会电子计算机和信息处理委员会发布的一套流程图符号基本一致．在

图 1-6 中(a)为矩形框,亦称“处理框”，表示一般处理功能;(b)为菱形框,亦称“判断框”,表示在几个可以选择的路径中,判断选择某一路径; (c)为两头尖框，表示循环; (d)为平形四边形框,表示输入/输出，本书用在平行四边形框中写入 I 或 O，分别表示输入(INPUT)或输出 (OUTPUT); (e)为圆弧边框,亦称“端框”，表示流程开始或结束; (f)为圆圈,表示连接点; (g)为双线矩形框,表示调用辅程序;(h)为指向线,表示流程的路径和方向.

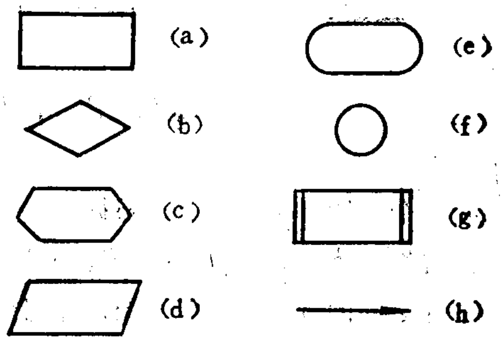


图 1-6 流程图常用符号

说明: (1)框内注释文字没有统一规定,以表达意思清楚为准. (2) 流程图符号中没有非执行语句可用的符号,通常也不允许把它们填入任何流程图符号中.

1.5 结构化程序设计简介

在 60 年代末以前,由于计算机的运算速度低,存储量小,评价一个程序的优劣总是把效率放在首位. 为了提高效率,程序员常常为减少一条语句或节约一个存储单元苦思冥想,结果程序越好越难读. 随着计算机科学的发展,一方面计算机运算速度和存储容量千百倍地提高;另一方面程序规模越来越大,验证和维护程序的费用不断上升. 因此人们评价一个程序的优劣,已不再把效率放在首位,而是把易读易验证和易维护放在优先的地位. 为此,荷兰学者戴克斯特拉(E. W. Dijkstra)提出了“结构化程序设计”(Structured programming). 简要地说,结构化程序设计是“按照一组能够提高程序易读性与易维护性的规则而进行程序设计的方法”. 从 60 年代末期到现在，结构化程序设计已发展成为一套系统的科学的方法，得到了软件工作者的广泛承认和欢迎.

“工欲善其事，必先利其器”，为了实现结构化程序设计，要选择合适的计算机语言. 基本 BASIC,FORTRAN 66,都不能适应结构化程序设计的要求,FORTRAN 77 也是非完全结构化的语言,即使是标准 Pascal——结构化程序设计语言,也不如 True BASIC 结构化程度高.

1.5.1 三种基本结构

1966 年，Bohm 与 Jacopini 在一篇文章中指出,任何程序的逻辑均可用顺序、选择、循环三种基本结构或它们的组合表示. 三种基本结构的流程图如图 1-7 所示.

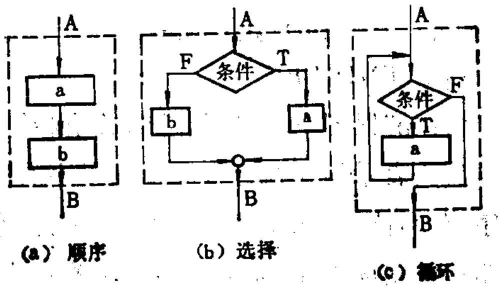


图 1-7 三种基本结构的流程图

(1) 顺序结构.

如图 1-7(a)所示,它有一个入口 A,一个出口 B. 在这个结构中的各块是按它们出现的先后顺序依次执

行的．这是一种最基本、最简单的结构．关于"块"的解释，待介绍了三种基本结构之后再作进一步描述．

(2) 选择(分支)结构．

如图 1-7(b)所示，它有一个入口 A，一个出口 B．选择控制结构执行过程是，如果条件成立(为真)，则执行 a 块，否则执行 b 块，再转到出口．总之，选择控制结构，只选择其中一个分支执行一次．

(3) 循环(重复)结构．

如图 1-7(c)所示，它有一个入口 A，一个出口 B．循环结构执行过程是：当满足给定的条件时，则重复执行 a 块，否则转到出口 B．从图 1-7(c)可以看出，循环是选择结构的一种特殊形式，其中一个分支趋向出口，而另一个分支返回结构入口，从而构成重复执行．

"块"可以是一个可执行语句(控制转移语句除外)，或一个空块(即无任何可执行的语句)，或三种基本结构之一．由于基本结构可以嵌套基本结构，所以，不难理解上述"任何程序的逻辑均可用三种基本结构或它们的组合来表示"．

1.5.2 结构化程序设计准则

一个结构化程序只能由以上三种基本结构组成，并应当满足以下条件：

(1) 单入口，单出口．

(2) 对每个结点(语句或块)，都应当有一条从入口到出口的通路通过它，即不允许有永远执行不到的可执行语句．

(3) 不包含无限循环，即执行的时间是有限的．

后两个条件也是传统的非结构化程序所要求的，所以结构化程序的特点在于，它的基本结构都能满足单入口、单出口的条件．

1.5.3 结构化程序设计方法

人们认识解决问题的方法，从逻辑上说，粗略地分为两种，一是从部分到整体，"自底向上"，即综合法；二是从整体到部分，"自顶向下"，即分析法．结构化程序设计方法是属于后一种，就是从问题出发，由粗到细，逐步细化，直到很方便的写出程序为止．概括地说，结构化程序设计方法就是"自顶向下，逐步求精"．

1.5.4 结构化流程图——N-S 图

前面我们介绍了一般流程图符号及使用方法，这种流程图有个指向线(即箭头)，使得流程图的画法很自由、灵活，非常适合非结构化程序设计使用，因而过去使用很广泛．它的缺点是：由于流程图的画法过于灵活，使得同一个解题过程(算法)可能会有多种画法，构成多种多样的流程图．这样，就使得流程图与程序之间的对应关系不那么直观了．

结构化程序设计方法提出之后，为了适应结构化程序设计方法的要求，于 1973 年由 Nassi 和 Shneiderman 首先开发了结构化流程图，这种框图又称为 N-S 图．本书使用的 N-S 图是在 Nassi 和 Shneiderman 框图的基础上形成的，它正确地表现了"每个基本块是单入口单出口"的结构化程序设计基本原则．N-S 图的优点是：画起来简便，而且和程序有良好的一致性．例如用

N-S 图表示图 1-5，如图 1-8 所示。

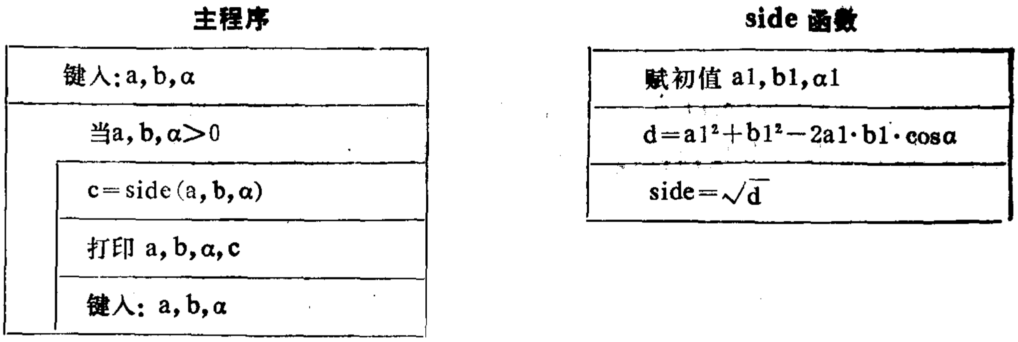
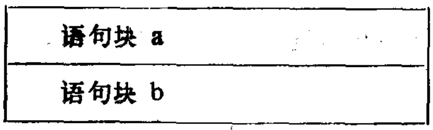


图 1-8　N-S 图

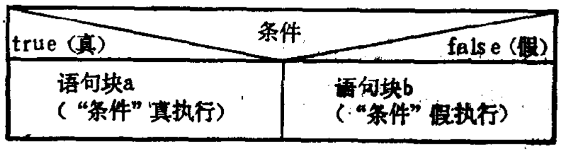
下面用与一般流程图对照的方法介绍几种 N-S 图，这些图将作为本书画 N-S 图的基本结构图.

(1) 顺序、选择和循环结构.

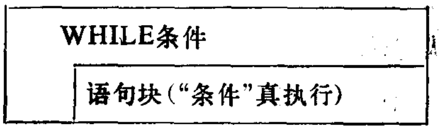
如图 1-7 所示的顺序、选择和循环结构对应的 N-S图，如图 1-9 所示.



(a) 顺序结构 N-S 图



(b) 选择结构 N-S 图



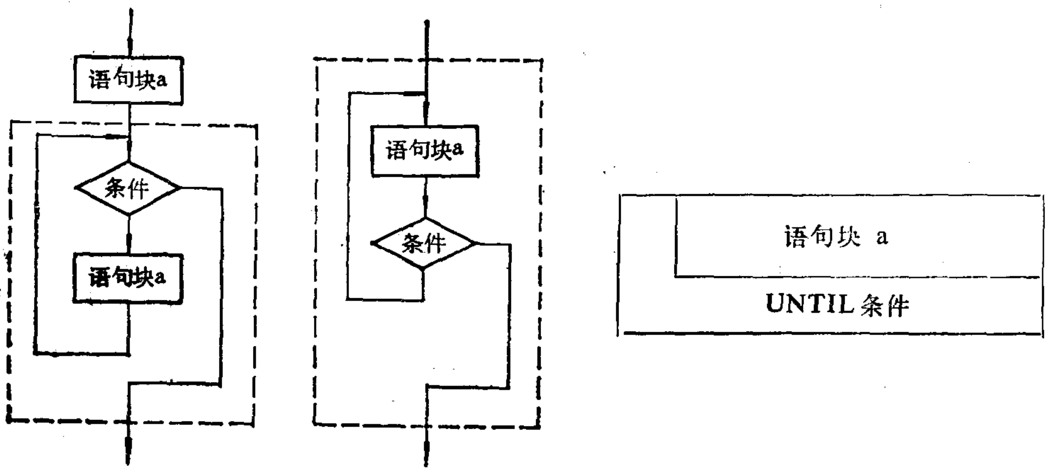
(c) 当型循环结构 N-S 图

图 1-9　三种基本结构的 N-S 图

在 N-S 图中，框的上端线为入口，框的下端线为出口，每一个矩形框只有一个入口和一个出口. 整个 N-S 图总是自上而下执行的. 图 1-9 (c) 是"先判断后执行"的重复结构，即当条件成立(为真)时，重复执行循环体(被重复执行的语句块)，否则不执行循环体，转至循环结构的下一个结构执行. 通常称这类循环为当型循环结构.

(2) 直至型循环.

图 1-10(a)是一个矩形框和一个当型循环结构的复合结构，而且矩形框和循环体是一样的，因而可以简化为图 1-10(b)表示. 这个结构是"先执行后判断"的重复结构，即重复执行循环体，直至条件成立(为真)时，结束循环转至循环结构的下一个结构执行，否则继续执行循环体. 通常称这类循环为直至型循环结构. 本书规定：它的 N-S 图如图1-10(c)所示.

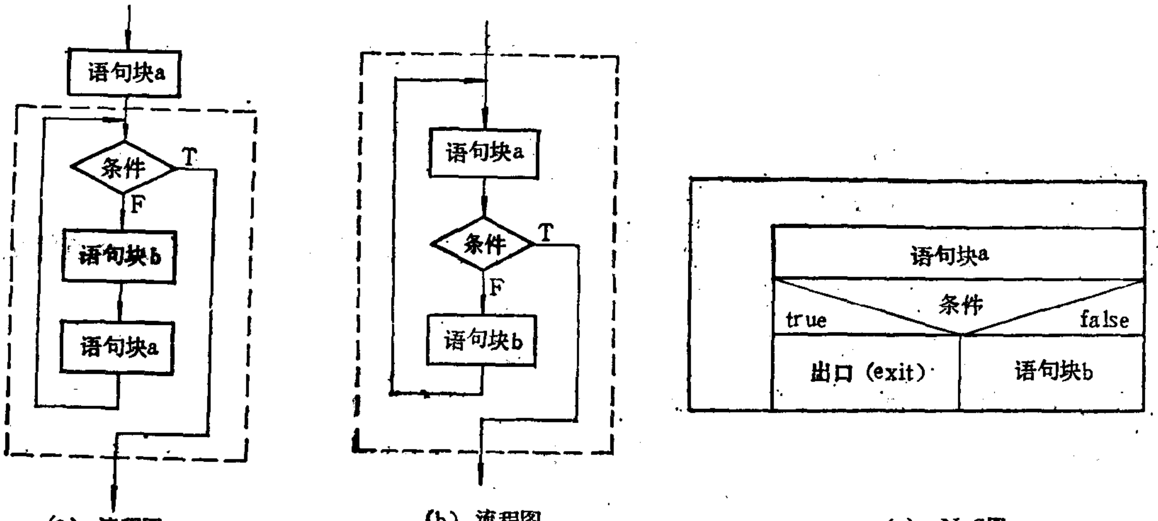


(a) 流程图　　(b) 流程图　　(c) N-S 图

图 1-10　直至型循环结构的流程图与 N-S 图

(3) 带条件出口的无限循环.

图 1-11(a)是一个矩形框和一个当型循环的复合结构，而且语句块 a 和循环体内的一部分语句块 a 一样，因而可以简化为图 1-11(b)表示，这个在无限循环中带条件出口的结构也是一种常用的结构. 本书规定：它的 N-S 图如图 1-11(c)所示.



(a) 流程图　　(b) 流程图　　(c) N-S图

图 1-11

例如，图 1-5 的主程序流程图就可以用带条件出口的无限循环结构表示. 如图 1-12 所示.

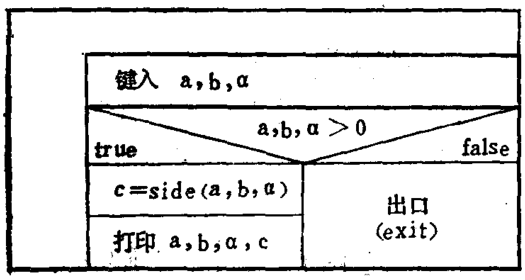


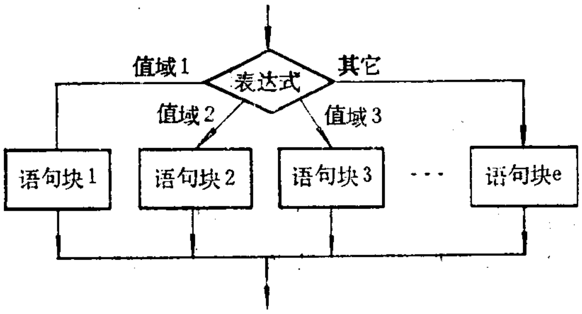
图 1-12

注意，这个"出口"规定是将控制转入循环结构的下一个结构执行．这个"出口"和循环正常结束出口是一致的．所以在循环中附加了这个条件出口语句之后，仍符合结构化程序设计方法"单入口单出口"的基本原则．

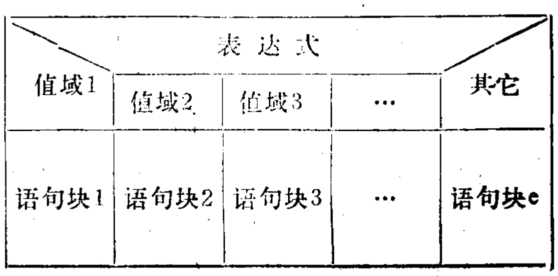
同样，在当型循环与直至型循环中都可以附加条件出口语句．这里不再赘述．

(4) 多分支结构．

图1-13(a)是一般流程图的多分支结构，根据表达式的当前值确定执行哪个分支．执行路径如图所示．这个结构是两个分支的选择结构的复合结构，可以用选择结构嵌套表示．这种多分支结构也是一种常用的结构，本书也选作基本结构，并规定它的N-S图如图1-13(b)所示．



(a) 流程图



(b) N-S图

图1-13 多分支结构

(5) 计数型循环结构．

本书也作为基本结构之一，留在5.3节介绍．

1.5.5 评价程序的基本准则

(1) 可靠性．数据类型要合理，语句要合法，程序要完整，逻辑要正确，经过认真测试和运行．这是最基本的要求．

(2) 可维护性．逻辑通顺、易读、易理解，便于修改扩充．

(3) 通用性．不但适用于某一个问题，而且适用于某一类问题．

(4) 高效率．运行速度快，占用存储单元少．

以上四点是有矛盾的，在实际编写程序时，需要综合考虑．

1.6 True BASIC 软件的优点

1.6.1 模块化结构化多功能的程序设计语言

(1) True BASIC 语言是符合大多数计算机科学家所倡导的"结构化程序设计"理论的一种模型．True BASIC 提供了便于描述选择和循环控制结构的结构化语句，以及主程序、子程序、函数和图画等程序模块，结构清晰．

(2) True BASIC 语言有高级彩色图形、动画和音响功能．

(3) True BASIC 语言包括了一些大型语言的功能．

a) 矩阵运算功能.

b) 外部辅程序库功能.

c) 多窗口显示功能.

d) 连接多个程序运行功能(可覆盖或不覆盖).

(4) True BASIC 和其它扩展 BASIC 比较，它保持了原 BASIC 的基本特点(小型、通用、会话、易学),语法规定自然合理,所以叫真(True) BASIC.

例如,在 IBM PC 机上实现,所需条件是:

a) 存储器:随机存储器(RAM)容量≥192K.

b) 软盘驱动器:至少一个双面磁盘驱动器.

c) 监视器:IBM 单色或彩色图形适配器.

d) 操作系统:MS-DOS 2.0 或更高的版本.

(5) True BASIC 与其它 BASIC 版本有良好的互通性.

1.6.2 简便宽松的操作环境

使用 True BASIC 软件编写程序,使你感觉简便、没约束,并随时可以得到系统软件的帮助(Halp).

(1) True BASIC 系统设有程序存储区和操作历程存储区，在显示器上对应有程序编辑窗口和操作背景窗口. 在编辑窗口对当前文件(内存中的程序) 可作全屏幕编辑;在背景窗口可作人机会话操作,还可随时查阅操作历程等.

(2) True BASIC 系统有解释和编译两种方式. 解释方式跟综能力强,编译方式效率高,可充分发挥两种方式的特长,即用解释方式调试程序,用编译方式产生目标程序,运行使用.

(3) True BAS1C 系统有较好的程序设计辅助功能.

a) 有删除行号、加行号和重编行号命令.

b) 有格式化命令,能把当前文件写为格式化的形式.

c) 有 DO 程序模块,可用它编写检查或修改当前文件的程序.

d) 有 WHEN ERROR IN—USE—END WHEN结构化控制语句,可用它编写“出错处理”程序段.

e) 有 PACKB 和 UNPACKB 辅程序,利用它们可以有效地利用存储空间.

(4) True BASIC 提供了 PEEK 函数和 POKE 子程序，可用它们直接观察和修改计算机内存字节,从而允许用户使用整个的存储器.

(5) True BASIC 支持通讯接口. 尽管各种计算机的通讯接口各不相同，但是 True BASIC 可以调用汇编语言写的子程序,因此,它支持各种计算机的管理通讯接口的子程序库.

习　题

1.1 何谓程序? 什么叫程序设计? 为什么要学习程序设计?

1.2 计算机的主要特点是什么? 它主要应用在哪些方面?

1.3 简要说明计算机由哪几部分组成？存储器的特点是什么？

1.4 编译程序的功能是什么？

1.5 流程图的作用是什么？

1.6 什么样的程序是结构化程序？结构化程序设计的基本方法是什么？

1.7 本书推荐了哪几种 N-S 图为基本结构图？

1.8 概括地说明 True BASIC 软件有哪些优点？

1.9 学习了本章之后有哪些体会？

第 2 章　True BASIC 语言初步知识

通过前一章的学习，对计算机的基本组成，以及 True BASIC 语言程序设计在计算机应用中的地位有了概括的了解．这一章介绍 True BASIC 语言的初步知识，包括字符集、数据、语句和True BASIC 程序块．这些是构成计算机语言的基本要素．

2.1 字符集与符号名

2.1.1 字符集

和自然语言不同，通常一种计算机语言只允许使用规定的标准字符，这有限个数字符的全体称为该计算机语言的字符集．然而，True BASIC 未作严格的限制，特别是在一个字符串或注释中只要是机器能识别的字符，True BASIC 都允许使用．美国国家标准代码——ASCII 字符集即在常见的计算机上都可以用的，也是 True BASIC 最基本的字符集，简介如下．

(1) 字母(letter)．

Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo Pp Qq Rr Ss Tt Uu Vv Ww Xx Yy Zz(大写、大写字母通用)．

表 2-1

字符	名称	字符	名称	字符	名称
+	加号	.	小数点	"	双引号
−	减号	,	逗号	?	问号
*	星号	;	分号	%	百分号
∧	箭号	:	冒号	$	美元符
/	左斜线	!	警叹号	@	圈a
\	右斜线	&	合并号	|	竖直线
(	左圆括号	>	大于号	␣	空格
)	右圆括号	<	小于号	_	下线符
[	左方括号	=	等号		
]	右方括号	～	波折号		
}	左花括号	`	左单引号		
{	右花括号	'	右单引号		

(2) 数字(digit).

0 1 2 3 4 5 6 7 8 9

(3) 特殊字符,如表2-1所示.

几点说明:

(1) 空格. 为了程序易于阅读,在书写程序时,可以在适当地方留一个或几个空格. 在书写时,常用␣表示,而在打印或显示时,空格就是空白,没有任何标记.

(2) 在 True BASIC 程序中, 一般不使用 ASCII 字符集以外的字符, 以便有较好的 可移植性.

(3) ASCII 字符集除上述可图视的图形字符外, 还有不可图视的控制字符. 在编写应用程序时,很少使用. 详见附录 B.

2.1.2 符号名(标识符)

在日常生活中,人们为区别各种东西,要给每个东西取个名字. 同样对程序中使用的各种变量、函数等也要分别取个名字. 在 True BASIC 中,它们都是用"符号名"标识的. 因此, 符号名是常用的术语.

(1) 符号名的形式.

True BASIC 符号名, 如图 2-1 所示. 符号名可包含字母、数字和下线符号(_). 最长可达 31 个字符.

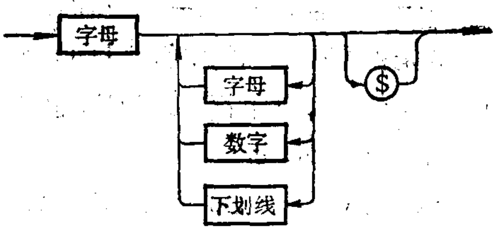


图 2-1 符号名语法图

几点说明:

a) 符号名必须是字母开头,中间不能有空格.

b) 在符号名中,大小写字母通用,视作一样.

c) 符号名的选择. 原则上说,只要符合符号名形式规定,程序员有权自由选择. 习惯是: 常用从简,专用从意,习惯相近,使用方便. 这样可提高程序的可读性.

例如,在数学中使用 a, b, c, π, I_{max}, 而在程序中可使用 a, b, c, PI, imax. 再如姓名、年龄、住址,可使用相应英语单词 name, age, address 或拼音.

(2) 保留字.

表 2-2 中的符号名,在 True BASIC 系统中已有特定含义, 这样的符号名称之为"保留字".

表 2-2

关键字	标准函数名		标准子程序名
ELSE	DATE	MAXNUM	DIVIDE
ELSEIF	DATE$	PI	PACKB
IF	DET	RND	POKE
NOT	EXLINE$	TIME	
PRINT	EXTEXT$	TIME$	
REM	EXTYPE		

注①关键字．True BASIC的每个语句都用特定的符号名表示，这些符号名称为关键字．关键字的含义多与英语单词含义一致．如 PRINT 是打印输出语句的关键字．但不是所有关键字都是保留字．

② 标准函数、标准子程序是 True BASIC 软件提供的程序．如 RND 是随机函数名，引用它即生成一个随机数．

2.2 数　据

我们常说：编写程序是为了处理数据．反过来说，数据是程序处理的对象．True BASIC 的数据分为两类：数值类型和字符串类型．每类又各有三种形式：常数、简单变量和数组变量．数组留在第6章讲述．

2.2.1 数值常数

数值常数在程序执行中，其值始终保持不变．

(1) 数值常数的形式．

True BASIC 数值常数如图 2-2 所示．数值常数是由符号部分，数字部分和指数部分组成．字母 E 表示以 10 为底．数值常数是十进制数值．

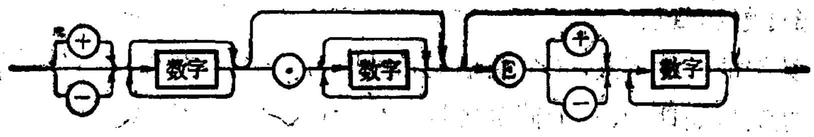


图 2-2　数值常数语法图

上述形式规定，使得书写数值常数相当自由，象数学上的整数、小数或指数形式都可以．但注意：10^7 写作E7 是不允许的，要写作 1E7，否则会误为符号名E7．另外，数值常数中间不允许有空格或逗号出现．

(2) 数值范围与精度．

一般地说，数值范围和精度是受硬件条件约束的．在 IBM PC 机上，如表 2-3 所示．

表 2-3

数值精度与范围	不带8087	带8087
数字精度	14位有效数字	16位有效数字
sin cos tan atn log exp ）精度	10位有效数字	16位有效数字
最小正数(eps(0))	1.11254E－308	2.22507e－308
最大正数(Maxnum)	3.59539E＋308	1.7979E＋308

若一个数的绝对值超过最大整数(Maxnum)，在程序运行时，将出现"overflow"(溢出)错．

若一个数的绝对值小于最小整数(eps(0)),机器会自动地用0代替,不给出错信息.

True BASIC 提供了两个标准函数可以获取机器的数的极限值: 函数 Maxnum 可得到最大值,函数 eps(0)可求出最小值.

2.2.2 字符串常数(字符串)

(1) 字符串常数的形式.

字符串常数如图 2-3 所示.

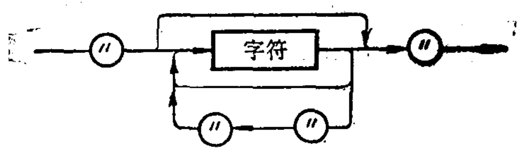


图 2-3 字符串常数语法图

例如(a)"No. Name. Result"

(b) " "

(c) "1987/11/26"

(d) "I say: ""This␣is␣a␣string."" "

(2) 几点说明:

a) 在字符串中,可以包含机器能识别的任何字符.

b) 在字符串中,大小写字母是不同的.

c) 在字符串中,空格和其它字符一样有效.

d) 在字符串中,一个双引号要用连续两个双引号表示. 如上面的(d)表示的那样.

e) 字符串长度,字符串中含有字符的个数叫字符串长度. 字符串没有任何字符时长度为0, IBM PC 机字符串的最大长度为 32767.

2.2.3 简单变量和赋值

简单变量是数据的一种形式,每个简单变量可以有一个值,在程序运行期间, 简单变量的值是可以改变的. 在数据处理中,一个简单变量相当于(代表)一个常数.

(1) 简单变量的标识与类型.

简单变量的名字(简称变量名),是用符号名标识的. 简单变量分为两类: 数值简单变量,它是用不带 $ 的符号名表示的;字符串简单变量(简称串变量),其符号名最后一个字符必须是 $.

例如,x x为数值变量,

x$ x$ 为串变量.

这样的类型说明叫隐式类型说明. True BASIC 数据类型简明扼要, 没有其它数据类型及其类型说明语句.

(2) 变量的赋值与变量引用.

在程序运行时,系统软件对程序的每个变量都要分配一个存储单元,用它存放变量的值. 从存储信息的角度来说,变量就代表存储单元. 所以,变量的值也是“取之不尽, 存之更新”. 就是说,对一个变量可以反复地取出旧值,一次又一次地用新值置换旧值. 计算术语把存入新值称为给变量“赋值”. 取变量的值称为“变量引用”. 变量引用仍保持变量的值不变, 相当于作了一次照象复制,而不会被“拿空”了.

在程序运行开始时,True BASIC 系统给所有数值变量赋值为 0, 给字符串变量赋值为空串(""). 在程序运行期间, 变量可以通过赋值语句、读入语句、键盘输入语句等多种方式改变

变量的值．下面先介绍赋值语句，其它留在以后再介绍．

2.2.4　变量赋值语句

一般形式：

LET　var＝expr

或　LET　var1，var2，…＝expr

其中　LET　为变量赋值语句的关键字．

var　为简单变量名．可以是数值类型或字符串类型．

expr　为表达式，即由常数、变量和运算符组成的式子．常数和变量是表达式的特殊形式．表达式的 True BASIC 表示法将在 3.1 节详述．

＝　为赋值号．

注意，赋值号两边数据类型要一致，否则会出错．执行它，首先计算表达式的值，然后把该值赋给赋值号左边的变量．即把值传送到变量的存储单元之中．

例如，

```
LET e=2.718281828
LET i,j=2
LET movth$="May"
```

2.3　语　　句

True BASIC程序是由若干条语句组成的．True BASIC语句很丰富，详见附录 C. True BASIC的每条语句都是由一个关键字开始的．

2.3.1　简单语句与结构语句

True BASIC 语句按组成结构分为简单语句和结构语句．简单语句由一条语句组成，占用一行，在程序中出现是独立的．结构语句由若干条子句组成，一个子句占用一行，在程序中各子句的出现是相互关联、彼此呼应的，要匹配使用，以表示一个完整的结构语句，或程序模块．

例如

- 简单语句：
  - 输入/输出语句
  - 赋值语句
  - 设置/询问特征值语句
  - 图形语句
  - 音响语句
  - 停、暂停语句
  - ……
- 结构语句：
  - 结构 IF 语句
  - 结构 CASE 语句
  - 结构 DO 语句
  - 结构 FOR 语句
  - SUB-END SUB
  - ……

2.3.2 可执行语句与非执行语句

True BASIC 语句按程序运行时是否产生操作，又可分为可执行语句(引起操作)；非执行语句(不引起操作).

如，LET 语句，PRINT 语句等为可执行语句；

DATA 语句，REM 语句，PROGRAM 语句等为非执行语句，只为系统程序提供信息.

可执行语句，在程序中出现要按照解题步骤描述的需要出现，要受严格的限制. 而非执行语句的出现，有的就比较自由. 如 DATA语句，是为读语句提供数据的，可以放在程序的任何位置. 有的要在指定位置出现. 如 PROGRAM 语句，用以标识程序开始，它永远放在程序开头，一个程序至多有一条.

2.4 源程序（程序）

2.4.1 源程序的结构

True BASIC 源程序是由一个主程序或一个主程序和若干个外部辅程序组成. 通常把程序中的主程序和辅程序都称为程序模块(或程序单位). 所以，True BASIC 属于模块化的程序设计语言. 每个程序模块可由简单语句、结构语句和内部辅程序组成.

例如，ex01_2程序是由一个主程序和一个外部函数组成的.

(1) 程序单位 { 主程序；辅程序 { 外部子程序；外部函数；外部图画 } }

(2) 主程序

```
PROGRAM   程序名
……
……  } 程序体
……
END
```

主程序从 PROGRAM 语句开始到 END 语句结束，结构完整清晰. True BASIC 为和 BASIC 兼容，规定也可以不写 PROGRAM 语句.

本书在前七章先讲主程序的设计及多种语句的使用，辅程序的设计留到第 8 章以后再讲.

2.4.2 源程序的书写格式

(1) True BASIC 源程序至少要有一个主程序. 程序运行总是从它的第一条可执行语句开始. 辅程序可有可无，根据需要而定.

(2) 一个简单语句占一行，“一行”以 enter 键为结束标志. 在一行内最多可写 32767 个字符. 一个结构语句可以占多行，详见各结构语句的形式规定或附录 C.

(3) True BASIC 程序可以没有行号. True BASIC 有很强的控制结构语句，可以不用GOTO和GOSUB语句,因此程序可以没有行号.本书例题均按结构化程序设计方法编写的程序,没有使用 GOTO 和GOSUB 语句. 但是,为了文字叙述和课堂讲解的方便，本书的例题又都给出了“行号”,实质上,它是行的序号、True BASIC 为了和基本 BASIC 兼容,也可以有行号. 注意,若用 GOTO 或 GOSUB 语句,则程序的每行都应有行号,行号次序要递增，在 1～999999 中任选.

(4) 注释. 用注释可以对程序的整体或局部作适当说明，增加程序的易读性. 注释可放在任一行上,如 REM 语句(ex01_2 的第 1 行)，或在一行的任何位置放一个惊叹号(!)，并在其后写一段正文作注释(如 ex01_2的第2, 8,10 行). 在惊叹号之后,写入任何“语句”都是无意义的.编译程序对注释不予理采,只是原样输入或输出.

(5) 本书源程序的书写格式.

a) 用缩进格式书写,嵌套层次分明.

b) 关键字用大写字母,符号名用小写字母,以便区别.

c) 程序名用 ex□□-□□,前面两个数字表示章,后面两个数字表示在章内程序的序号,以便查找.

希望读者在书写程序时,也要注意养成良好的习惯.

习　题

2.1 下列字符序列中,哪些可以作符号名? 哪些不可以? 为什么?

X	X1	1X	Z001	SUM	π
MA *	DJS−8	MQX	M1,2	M(1)	−MIN
Z−80	string	α	BATA	β	Pi
X$	J_1$	D$1	D. D. T.	root/1	GM(3,2)

2.2 True BASIC 有几类数据? 各类有哪几种形式?

2.3 举例说明什么叫可执行语句? 什么叫非执行语句?

2.4 True BASIC 程序的书写形式有哪些基本规定?

2.5 True BASIC 语言有哪几种程序模块(程序单位)?

2.6 下述哪些是合法的常数? 哪些是非法的常数?

E4	10,000	1.05	13.45	−7e2
5.3×10^{3}	−1.05	+100	0.5E−3	+0.5E3
"Abc"	"con't"	""	"F$="ON"or"OFF""	"plase again."

第 3 章 顺序程序设计

顺序结构如图 1-7(a)所示. 如果一个顺序结构的每一个块都能用一个语句表示，则称该结构为简单顺序结构. 表示简单顺序结构的语句序列就叫顺序程序段. 这一章将介绍仅包含顺序程序段的程序设计. 顺序程序的执行是从第一个可执行的语句开始，依次一个语句接着一个语句地执行，直至结束语句为止. 顺序执行是程序运行的基本规则，除 IF，DO， FOR 和 STOP 等控制语句外，其它的可执行语句都是顺序执行的语句. 如 LET 语句就是顺序执行的语句. 本章将引入 INPUT，PRINT 等语句和表达式. 通过键盘输入、屏幕显示，可实现人机对话式的算术计算和字符处理.

3.1 表 达 式

在 True BASIC 语言中，可以用表达式描述算术计算、字符处理和逻辑运算，分别称为数值表达式、字符串表达式和逻辑表达式. 逻辑表达式留到第 4 章讲述.

表达式是由常数、变量、函数及操作符组成的式子.

3.1.1 算术表达式

(1) 算术操作符:

数值运算	加	减	乘	除	乘方	括弧
True BASIC 算术操作符	+	-	*	/	^	( )

例 pi*r*r　　表示 πr^2

b*b-4*a*c　　表示 b^2-4ac

a*a+b*b-2*a*b*cos(alpha)　　表示 $a^2+b^2-2ab\cos\alpha$

其中 pi，cos(x) 都是系统提供的内部函数.

任何复杂的计算，都将用这五种基本运算表示. 例如

$$e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots$$

$$\ln x=(x-1)-\frac{(x-1)^2}{2}+\frac{(x-1)^3}{3}-\frac{(x-1)^4}{4}+\cdots$$

$$\sin x=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}-\cdots$$

由此可见，指数、对数和正弦函数都可以按上式转化为基本运算．推而广之，求方程式的根，计算定积分的值，解线性方程组等等，都可以按照一定的算法，通过基本运算得到答案．

注：① a∧b. 若 b 为整数，如 b=3 化作 a*a*a；若 b 为非整数，化作 exp(b*log(a))，因为 $a^b=e^{\log a^b}=e^{b\cdot\log a}$=exp(b*log(a))． 所以当 b 为非整数时，a 必须大于 0，否则会给出错误信息．

② 表达式中的每一个字符都必须写在同一行上，不准出现“上角标”或“下脚标”．

(2) 运算优先规则：

a) 先乘除，后加减(*，/ 优先于+，-)．

b) 乘方优先于乘和除．

c) 函数优先于乘方．

d) 括号最优先．在 BASIC 中括号无大、中、小之分．一律用（ ），可以在（ ）内再套用（ ）．内层优先于外层．

例如
```
2*sin (3*x) ↑5/2-8
↑  ↑    ↑    ↑ ↑ ↑
④  ②    ①    ③ ⑤⑥
```

小圆圈内的数字表示运算顺序，也就是说先进行括弧内的运算，再进行函数 sin 运算，求出 sin(3*x)后才进行④运算，乘和除同等级，按先左后右运算，最后减 8．归纳起来，用图表示优先顺序如下：

（ ） ⟶ 函数 ⟶ ∧ ⟶ *，1 ⟶ +，-

(3) 系统内部数值函数．

True BASIC 系统提供了许多内部函数，简称系统内部函数．这里将说明以数值作为自变量，并且返回结果也是数值的函数，如表 3-1 所示．第 3 章中将说明返回值为逻辑值的函数．第 6 章中将说明那些处理数组和矩阵的函数．

表 3-1

名　称	函数形式	含　　义
绝对值	abs(x)	|x|
取整	int(x)	取≥x 最大整数
取整	round(x)	四舍五入取整
取近似值	round(x,n)	取n位小数后近似值
自然对数	log(x)	$\log_e x$
2为底对数	log2(x)	$\log_2 x$
10为底对数	log10(x)	$\log_{10} x$
指数	exp(x)	e^x
最大数	max(x,y)	取x，y 之中最大值

名　　称	函数形式	含　　义
最小数	min(x,y)	取x,y之中最小值
平方根	sqr(x)	$\sqrt{x}$
结果精度	eps(x)	max(x−x′,x″−x,sigma) 这里x′和x″分别是x前趋和后继,sigma是能表示的最小正数
机器最大数	maxnum	IBM PC机值为3.59539E+308
伪随机数	rnd	0≤rnd<1 如果在使用rnd之前,执行RANDOMIZE语句,则将无法预言rnd的结果
取余	mod(x,y) remainder(x,y)	mod(x,y)=x−y* int (x/y) 其结果在0与y之间 余数与x符号一致而结果未必在0与x之间
取符号	sgn(x)	返回1或0或−1,和x符号一致
弧度变角度	deg(x)	x弧度转为角度
角度变弧度	rad(x)	x度转为弧度
求点(x,y)极角	angle(x,y)	x,y不能同时为0,−pi<angle(x,y)≤pi
正弦	sin (x)	sin x
余弦	cos(x)	cosx
正切	tan(x)	tanx
反正切	atn(x)	$\tan^{-1}x$
圆周率(π)	pi	3.1415926…

几点说明:

① n,x,y为数值表达式. a$,b$为字符串表达式.

② 如果三角函数的角没作说明，按系统规定是以弧度为单位. 如果初始数据以角度为单位，一种方法是可用角度变弧度函数化为弧度. 另一种方法是可用如下语句说明三角函数的角是以角度为单位.

OPTION ANGLE DEGREES

还可以用如下语句重新说明三角函数的角是以弧度为单位

OPTION ANGLE RADIANS

③ 除表3-1的系统内部函数外,还有一个系统内部子程序,求余求商:

CALL Divide(x,y,q,r)(其中q=int(x/y),r=mod(x,y))

3.1.2 字符串表达式

在第2章已讲过字符串常数、字符串变量和赋值. 例如:

LET old1$="This is a car."

LET old2$="That is a cart."

为了表示字符串变量值的一部分,True BASIC 引入了子串的概念. 子串形式是

Sname$[b,e]

它表示 Sname$ 的第 b 到第 e 个字符组成的串.

(1) 子串的赋值.

用对子串赋值可以改变字符串变量的值.

LET Sname$[b:e]=expr$

它的作用是用 expr$ 代替 Sname$ 的第 b 到第 e 个字符. 这里字符个数不一定相等,所以对子串赋值可以修改、插入或删除字符串变量的部分的值.

例如:

LET old1$[1:7]="I␣have"

这时 old1$ 的值为"I␣have␣␣a␣car."

若 b<1,则用 1 代替;

若 e 超过串长度,则用串长度代替;

若 b>e,则在第 b 个字符前插入 expr$.

(2) 合并.

合并是将字符串常数、字符串变量、串函数或子串连接起来的运算,合并运算符用 & 表示.

例如:

LET New$="This"& old2$[5:15]

New$ 的值是"This is a cart."(这是一辆马车.)

LET A$="Those are"& old1$[11:13]&"S".

A$ 的值是"Those are cars."(那些是汽车.)

(3) 字符串表达式的子串.

字符串表达式的子串是用来取字符串表达式值的一部分的运算. 形式是

(字符串表达式) [b:e]

例如:

LET B$=("That is a "& old2 $ [11:14]) [9:14]

B$ 的值为"a cart".

LET C$="AMA"&("ZED")[2:3]

C$ 的值为"AMAED".

LET D$=("AMA"&"ZED")[2:5]

D$ 的值为"MAZE".

注意,用于标识子串的[ ]可用( )代替. 实际上,美国国家标准 BASIC 要求使用圆

括号.

(4) 系统内部字符串函数　如表 3-2 所示.

表 3-2

函　　数	含　　　义
len(a$)	a$中字符的个数
lcase(a$)	a$字母皆变为小写字母
ucase(a$)	a$字母皆变为大写字母
trim$(a$)	去掉a$开头和结尾的空格
ltrim$(a$)	去掉a$开头的空格
rtrim(a$)	去掉a$结尾的空格
ord(a$)	a$为一个字符,求ASCII代码,与chr 函数互逆
chr(n)	chr$(mod(n,256)),求代码的字符
num(a$)	把字符串变成一个数,与num$函数互逆
num$(n)	把一个数变成IEEE 8字节表示法的数字字符串
str$(n)	把一个数变成十进制数字字符串,可阅读
val(a$)	把一个十进制数字串变成数,与str$函数互逆
pos(a$,b$)	b$在a$中第1次出现时,b$第1个字符在a$中的位置
pos(a$,b$,n)	b$在a$中第n次出现时,b$第1个字符在a$中的位置
repeat$(a$,n)	结果是n个a$的并
date	将当前日期用一个数返回,用yyDDD编码,如85033表示1985年的第33天,若无此功能,返回－1
time	将当前的时间用一个数返回,以午夜后的秒表示,如60,表示在午夜后1分,若无此功能,返回－1
date$	将当前日期以美国国家标准形式"yyyymmdd"表示,如"19850214",表示1985年2月14日
time$	将当前的时间以美国国家标准形式"hh:mm:ss"表示,如"13:45:02",表示13时45分2秒

此外还有

using$(format$,expr1,expr2,…)

可在PLOT 语句,WRITE 语中使用,和在 PRINT 语句中一样,详见 3.3 节.

3.2 PRINT 语　句

PRINT 语句可以把字符串和数值显示到屏幕上，或输出到磁盘上的正文文件中(详见第

9 章).

3.2.1 数值和字符串的显示

(1) True BASIC 数值显示规定.

a) 符号占一位，正数和零前面输出一个空格. 负数前面输出负号. 数字末尾加一个空格，以便与后继输出断开.

b) 数值能按整数显示的按整数显示,最多不超过 12 位.

c) 数值一般以小数形式显示,有效数字等于或小于 6 位(小数部分末尾的零不被显示). 如果无法用小数形式显示,则自动按科学表示法显示,即□d. ddddde±dd 形式.

例如, −2.66667e−78 (数字过小)

1.34e+19 (数字过大)

注意: 如果需要有效数字大于 6 位精确数字，或需特殊指定格式的输出，可用 PRINT USING 语句(见 3. 3 节).

(2) 字符串显示规定.

字符串总能准确的输出. 把字符串或数字输出到屏幕上，除规定它们的显示形式之外，还要确定位置. True BASIC给出了逗号区域定位法和TAB 定位法.

3.2.2 逗号区域定位法

(1) 边界与区域.

为了确定输出字符串和数字在窗口上的位置，系统规定了显示的边界和区域，如图 3-1 所示.

边界是 1 到 80 列(字符). 分 5 个区域,每个区域宽为 16 个字符. 一行输出最多是 80 个字符，如果超过边界,余下部分移到下一行继续输出.

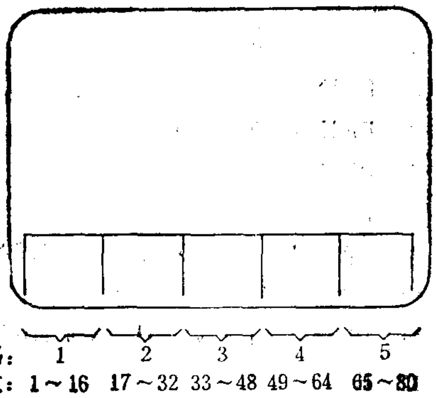


图 3-1 屏幕显示区界

边界和区域宽度可以用如下语句重新设定:

SET MARGIN expr1(设定边界)

SET ZONEWIDTH expr2(设定区域宽度)

其中 expr1, expr2 为数值表达式, expr1, expr2≥1 且 expr1≥expr2.

边界和区域宽度也可用如下语句询问:

ASK MARGIN var1

ASK ZONEWIDTH var2

其中 var1 和 var2 为数值变量. 执行上述 ASK 语句，系统会自动地把当前的边界和域宽值赋给变量 var1 和 var2.

(2) 逗号与分号定位.

确定输出的数字和字符串出现在屏幕的哪个区域,还要用 PRINT 语句规定. 常用的PRINT 语句格式有如下四种:

a) **PRINT**

执行它,回车换行,置光标于下一行开始.

b) **PRINT expr**

执行它,输出 expr 值后回车换行.

c) **PRINT expr1,expr2,**

执行它,输出 expr1 在第一区,输出 expr2 在第 2 区,并置光标在第 3 区开始——称按区域格式输出.

d) **PRINT expr1;expr2;**

执行它,输出 expr1 和expr2 后,置光标于 expr2右边一格. 输出字符串首尾不留空格,数值末尾留一个空格,输出项的值自左至右顺序输出——称按紧凑格式输出.

上述输出项 expr 可以是数值表达式或字符串表达式. 包括常数、字符串、变量、函数和表达式. 显示在同一行上的字符序列叫一个"记录". 每个记录是以回车换行为结束标志的.

例如:

```
1   PROGRAM ex03_1
2   REM PRINT with commas and semicolons
3   !
4   PRINT 10,20,
5   PRINT 30,40
6   PRINT 10;10+10;
7   PRINT 30,40;
8   PRINT
9   PRINT" ","NO.","Name","Age"
10  PRINT " ",1001,"lin ping",1987—1969
11  PRINT "I say:""He is a student."""
12  END
```

运行结果:

```
10             20          30          40
10  20   30    40
               NO.         Name        Age
               1001        lin ping    18
I say:"He is a student."
```

注意,第 4 行输出 10,20 之后以逗号结束,所以第 5 行输出的 30,40 仍显示在同一行上,然后再换行. 第 6,7 行四个数值显示在同一行上,因 40 之后是分号,所以光标停在 40 之后. 当执行第 8 行时,才回车换行,光标移到下一行的起点.

3.2.3 tab 定位法

tab 用于 PRINT 语句中,用它将光标向前(右)移到一个新的位置,多用于造表. 格式是

tab(expr)

其中 expr 为数值表达式.

执行 **tab**, 首先计算 expr 的值,然后四舍五入取整,它表示以当前行的第 1 列为起点,将光标移到新位置(列).

若 **tab** 以分号结束(**tab(expr);**),光标位置决定于 expr 的值:

如果 expr>边界,则光标移到下一行.

如果 expr<光标当前位置,则光标移到下一行对应位置.

如果 **tab** 后没有输出项,则光标移至下一行开始.

若 **tab** 以逗号结束(**tab(expr),**), 则光标按 expr 移动后, 将继续向前移至下一个区域的起点.

例如:

```
PROGRAM ex03_2
REM     phone list
!
PRINT "Name", tab(30);"Number"
PRINT "Zhan Ping";tab(30);"401.5599"
PRINT "Lu Jian";tab(30);"87.5831"
END
```

运行结果:

```
Name                                    Number
Zhan Ping                               401.5599
Lu Jian                                 87.5831
```

3.3 PRINT USING 语句*

PRINT USING 语句的作用与一般 PRINT 语句不同. 由于 PRINT USING 语句不受显示区域的限制,仅按照定义的格式输出,所以要小心地定义输出格式. 因格式规定较为烦琐, 难于一下学会. 本节以列表的方式给出多种使用形式,供读者模仿使用.

True BASIC 的 PRINT USING 语句的两种格式是:

PRINT USING format$:iteml, item2……

PRINT USING format$:iteml, item2……;

其中 format$ 是字符串表达式, 表示同一行的输出项是什么样的一种格式. 每一个输出项按照输出字符串表达式定义的格式显示输出. 如果语句以分号结尾,光标停留在输出项的结尾处. 否则它移到下一行的开始位置.

在 format$ 中,可以包括文字字符和格式符号. 当文字字符出现在格式符号串(format$)中时,则这些文字将原样显示输出, 而表示格式的符号并不被显示输出,它们只是用来控制输出项的格式. 每一个要输出的项目总是严格地按照格式项描述的那样输出. 格式符号由如下字符

定义:

〈 〉	+	−	$	*	%	#	,	.	∧

如果 PRINT USING 语句中的输出项目比较多，则多出来的输出项按照从头开始的格式项输出．如果格式项比输出项多，剩下的格式项无效．然而在 format$ 中的格式项后面的文字字符总是要输出的．

以后将提供更详细的把数值和字符串按格式输出的使用说明．这里给出几个使用 PRINT USING 的例子．

语　　句	输　　出
print using" Temperature=##":temp	Temperature=35
print using"+#·#on$#,###·##":i,d	+9.5on $3,527.30
print using"##·##∧∧∧∧∧years":age	23.01e+013 years
print using"License<##%%%%%%":s$,n	License NH 074362
print using"## and":a,b,c	12 and 13 and 14 and
print using"%% plus%%":3	03 plus

3.3.1 PRINT USING 和数值

数值输出一般有比较复杂的格式定义串．用于数值输出的格式字符有下列几种：

a) 首部字符：

+ 在数值前面显示‘+’或‘−’．

− 在数值前面显示空格或‘−’．

$ 在数值前面显示‘$’．

b) 数值字符：

* 将前置零作为‘*’显示．

% 将前置零作为‘0’显示．

将前置零作为空格显示．

c) 其它字符：

, 将逗号插入在数字中适当的位置．

. 小数点定位显示．

∧ 显示指数部份．

在每一个格式项中，至少有一个数值格式字符．这些字符用来描述输出数值的格式．不能将不同的数值字符在一个格式项中混合使用．象"####"描述了四个前置空格的输出格式，而"%%%%"描述了四个前置零的输出格式，但"%%##"却是不合法的．

(1) 简单数值格式．

如果数值是负的，它的负号占一个字符(正数和零格式后不带符号)．在一些简单数值格式中，带有小数的数值取整，如表 3-3 所示．

表 3-3

数	格 式 项	结 果
1	# # #	1
32	# # #	32
−32	# # #	−32
99.9	# # #	100
1	% % %	001
−1	% % %	−01
1	* * *	**1
−1	* * *	−*1
32	* * *	*32

(2) 小数点格式．

如果想显示小数部分，就可在数值型字符中加一个小数点，小数部分就按所提供的位数显示(只允许有一个小数点)．如表 3-4 所示．

表 3-4

数	格 式 项	结 果
.543	###.##	.54
1	###.##	1.00
12.5	###.##	12.50
333.666	###.##	333.67
.543	%%%.%%	000.54
1	%%%.%%	001.00
12.5	%%%.%%	012.50
333.666	%%%.%%	333.67

(3) 起始符号格式．

为进一步控制数值符号的输出或前置一个 $ 符号，可以加入一种或二种首部字符，第一个符号可以重复，在输出时可能转换成前置空格(如果必要，这些符号可用来显示数字)，如表 3-5 所示．

(4) 带逗号的格式．

在格式项的数值型字符中，可以插入一些逗号．如果输出结果在与格式项中逗号相应的位

表 3-5

数	格式项	结果
32	+++#	+32
32	---#	32
-32	-###	-32
-32	---#	-32
32.3	$$$#	$ 32
32.3	$$$$$##.##	$ 32.30
32.3	+++%%%%.%%	+0032.30
32.3	$******.##	$ ****32.30
32.3	$$$$$+#.##	$ +32.30
32.3	+++++$#.##	+$ 32.30

置上两边都有数字，True BASIC 在这些位置上显示逗号(否则逗号就会变为空格输出)．如表 3-6 所示．

表 3-6

数	格式项	结果
32	##,###	32
1432	##,###	1,432
1e6	##,###,###.##	1,000,000.00
1432	$ #,###,###.##	$ 1,432.00

（5）科学表示法格式．

最后，可以用科学表示法显示输出数字．使用'∧'符号表示指数部分．指数部分必须在格式表达式的末尾，并且必须至少有 3 个'∧'符号，以便为'e+'或'e-'留出空间．数字被指定的空间所限定，如表 3-7 所示．

表 3-7

数	格式项	结果
1.2e3	###∧∧∧	120e+1
1.2e3	#.##∧∧∧	1.20e+3
1.2e3	#.##∧∧∧∧∧	1.20e+003
1.2e3	##.####∧∧∧	12.0000e+2
1.2e3	-#.####∧∧∧	1.2000e+3
-1.2e3	-#.####∧∧∧	-1.2000e+3

(6) 溢出格式.

如果数值比格式串所限定的空间大,则全部输出结果由星号代替. 如表 3-8 所示.

表 3-8

数	格式项	结果
100	##	**
−50	##	**
100	$$#	***
100	#.#	***
1e100	#∧∧∧	****

3.3.2 PRINT USING 和字符串

字符串可以由任意数值字符格式定义显示输出. 当字符串小于格式项定义的长度时,由空格充填在字符串的两边,使输出结果显示在中间的位置上(如果必要,右边的空格比左边的空格多一个). 如果字符串比定义的长,全部输出星号(*).

可以在格式项前加"〈"符号,使输出字符串左对齐;或者加"〉"符号,右对齐. 一个字符串的格式只能有一个"〈"或"〉".

表3-9是几个输出字符串格式的例子,分别为在中间显示,靠左边显示,靠右边显示或域溢出.

表 3-9

符号串	格式范围	结果
x	####	x
xy	####	xy
xy	<###	xy
xy	>###	xy
cash	$ ###	cash
dollarg	$ #.##	*****

3.4 打 印 输 出

True BASIC 程序的打印输出与屏幕显示输出形式一样. 具体方法是:

(1) 用如下语句打开打印机通道:

OPEN #expr:PRINTER

其中 expr 为算术表达式,其值为通道号,在 1～1000 之间.

(2) 使用PRINT语句、PRINT USING语句、MAT PRINT语句进行打印输出时，要在PRINT之后加上#expr:.

注意:在运行程序之前，务必接通打印机电路，否则会显示出错信息. 此外，打印是以记录为单位的，即直至出现回车换行才打印输出.

3.5 INPUT 语 句

用INPUT语句可以从键盘上输入常数，给对应变量赋值.

3.5.1 常用的INPUT语句格式

INPUT var1, var2, …varn

INPUT PROMPT expr$:var1, var2, …, varn

其中var是串变量或数值变量.

expr$是字符串表达式，作为待输入项的提示. 执行INPUT语句，如果没有提示部分(PROMPT expr$)，会在光标当前位置显示一个问号，等待键入. 这时用户要按输入项的顺序依次键入相应的值，值与值之间用逗号隔开. 如果有提示部分，首先会从光标当前位置起显示字符串(无问号)，等待键入. 用户回答方式和上述一样.

几点说明:

a) 光标当前位置是由上一个PRINT语句定位的. 用"PRINT"语句可把光标移至新行的第1列.

b) 如果不能将所有输入数据项写在一行上，则可在任一数据项之后键入逗号后回车换行.

c) 键入字符串是不必用双引号括起来的. 但是，如果字符串中含有逗号或双引号，或首尾有空格，则必须在字符串的两端加上双引号，而且串里的一个双引号字符要用两个双引号表示.

d) 空格需要用" "表示.

例如:

```
1  PROGRAM ex03_3
2  REM     ask for string
3  !
4  PRINT
5  INPUT prompt "string$ ":string$
6  PRINT    string$
7  END
```

运行结果:

```
string$  'He is called,↵
Badly formed input line.  Please retry from bad item.
? ""Zhang 3"" "↵
Badly formed input line.    Please retry from bad item.
```

```
? "He is called""Zhang 3"" "↵
He is called "Zhang 3"
```

这里输入的字符串含有双引号，每个双引号用两个双引号表示，而且在串的两端加上了双引号(注意:若一个字符串较长用一个逗号结束后回车换行，表示字符串输入未完，则是一种坏的输入，将显示"?"，需重新输入一个字符串).

注意:

a) 输入数据与变量类型要一致，否则出错.

b) 输入数据过大会出现溢出错误信息，需再输入.

c) 输入数据个数多于变量个数，一般多余的数据将被忽略. 如，ex03_3a. 如果输入表以逗号结束，则多余的数据项将留给下一个 INPUT 语句读入. 如 ex03_3b.

例如:

```
PROGRAM ex03_3a
INPUT a,b
INPUT c
PRINT a,b,c
END
? 1,2,3
Too many input items. Excess ignored. (输入项过多，超过被丢掉)
? 4
 1                2             4
```

```
PROGRAM ex03_3b
INPUT a,b,
INPUT c
PRINT a,b,c
END
 ? 1,2,3
 1                2             3
```

3.5.2 LINE INPUT 语句(行输入语句)

行输入语句能输入完整的一行字符，赋给一个字符串变量. 这一行字符可以包含空格、逗号、双引号等，逐个输入不作任何检查.

LINE INPUT 语句常用形式是:

LINE INPUT var1$,var2$,…

LINE INPUT PROMPT expr$:var1$,var2$,…

其中 expr$ 是字符串表达式，作提示信息.

PROMPT expr$ 为提示部分.

var$ 为字符串变量.

执行 LINE INPUT 和执行 INPUT 语句相似，不同的是 LINE INPUT 语句的每一个变量要输入一行字符. 行输入语句的输入项不准是数值变量.

用这个语句输入表头、标题或记录是非常方便的. 例如:

```
PROGRAM ex03_4
LINE INPUT line$,title$
PRINT
PRINT line$
PRINT title$
PRINT line$
END
```

运行结果:

```
?           ------------------------------------
?           |       Table of the result        |
            ------------------------------------
            |       Table of the result        |
            ------------------------------------
```

3.6 顺序程序应用举例

例 1 有四个并联的电器设备,电阻分别是 100, 150,240,360,电路总电压为 220,试计算总电流.

解 并联电阻为:

$$\frac{1}{R}=\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_4}$$

并联电流为:

$$I=U/R=U\left(\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_4}\right)$$

可编计算例 1 的程序如下:

```
PROGRAM ex03_5
PRINT "r1,r2,r3,r4"
INPUT r1,r2,r3,r4
LET u=220
LET i=(1/r1+1/r2+1/r3+1/r4)*u
PRINT "i=";i
END
```

运行结果:

```
r1, r2, r3, r4?
? 100,150,240,360
i=5. 19444
```

该程序可以用于计算任意个并联的家用电器的总电流.

例 2　任意给定 a, b 二值，按下式计算 x, y 的值.

$$x=(a^2+b^2)^3, \qquad y=(a^2+b^2)^4$$

解　下面给出了两种程序.

程序 1:

```
PROGRAM ex03_6a
INPUT prompt "a,b?":a,b
LET x=(a*a+b*b)^3
LET y=(a*a+b*b)^4
PRINT "x=";x,"y=";y
END
```

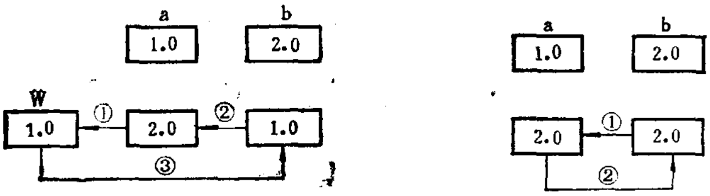
程序 2:

```
PROGRAM ex03_6b
INPUT prompt "a,b?":a,b
LET wk=a*a+b*b
LET x=wk^3
LET y=x*wk
PRINT "x=";x,"y=";y
END
```

程序 1 作了 9 次乘法运算，而程序 2 只作了 5 次乘法运算，明显地提高了效率. 原因是，在程序 2 中设置了一个临时变量(计算机术语叫“工作单元”). 设置工作单元，提前计算公用部分，这是程序设计中常用技巧之一.

例 3　交换变量 a 和变量 b 的值，简称变量交换，它是基本程序设计方法之一.

解　交换两个变量的值需要第三个变量，即需要设置新的工作单元 w，如图 3-2(a) 所示. 如果不设置新的工作单元，结果将是错误的. 如图 3-2(b) 所示.



(a) 正确的交换　　(b) 错误的交换

图 3-2　变量交换

依据图 3-2(a) 可编程序如下:

```
PROGRAM ex03_7
INPUT prompt "a,b?":a,b
LET w=a
LET a=b
LET b=w
PRINT "a=";a,"b=";b
END
```

例 4 试编一个程序，求一元二次方程

$$ax^2+bx+c=0$$

的两个实根，并用 a=1, b=3, c=2 验算之.

解 从代数学知道：

$$x=\pm\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

可编程序如下：

```
PROGRAM ex03_8
INPUT prompt "a,b,c?":a,b,c
LET d=sqr (b*b-4*a*c)
LET x1=(-b+d)/(2*a)
LET x2=x1-d/a
PRINT "x1="; x1,"x2="; x2
END
```

运算结果：

```
a,b,c? 1,3,2
x1=-1            x2=-2
```

习 题

3.1 把下列数学式写成 True BASIC 算术表达式：

(1) $\frac{a}{bc}$

(2) $\frac{x+2}{y-4}$

(3) $\frac{ax-by}{ax+by}$

(4) $a+\frac{b}{c+d}$

(5) $(x+y)^{\frac{5}{2}}$

(6) $\left(\frac{x}{y}\right)^{r-1}$

(7) $1-\frac{x^2}{2}-\frac{x^4}{24}-\frac{x^8}{720}$

(8) $(x+\sqrt{x^2-1})^n$

(9) $\cos\frac{y}{\sqrt{x^2+y^2}}$

(10) $\log|\sin x+\cos x|$

(11) $x^3-\frac{\sin(a^5-b^3)}{7.3\times10^{-8}-c}$

(12) $\cfrac{a_0}{x+\cfrac{a_1}{x+\cfrac{a_2}{x+a_3}}}$

3.2 按区域格式打印出下述函数的值：

Pi, Maxnum, eps(0), sin 45°, cos π/2, Rnd 和 Rnd.

3.3 设当前存储单元的内容如下所示：

A [0.3]　　B [3.1]　　C [5.0]

执行下列赋值语句：

```
LET A=37/c
LET B=A+1.6
LET D=B↑2
```

之后，变量 A，B，C，D 存储单元的内容是什么？

A[　　] B[　　] C[　　] D[　　]

3.4 计算坐标分别为(x_1,y_1)，(x_2,y_2)的两点之间的距离，两点的坐标值可以键入，求两点间距离的公式为 $l=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$.

3.5 已知地球半径 6371km，试编写一个程序，近似地计算赤道长度以及地球的面积和体积.

3.6 若给定圆柱半径和高，试编一个程序，计算圆柱的底面积、柱面积和体积.

3.7 已知某人马拉松跑的成绩是 XH 时 XM 分 XS 秒，计算它每小时平均跑多少公里？打印姓名和成绩.

3.8 某人工资若干元，问他可领 10 元、5 元、2 元和 1 元最少各几张？共多少张？

3.9 计算某年某月某日是星期几？

［提示］计算公式：

$$s=y+\frac{y}{4}-\frac{y}{100}+\frac{y}{400}+c$$

其中 y 为公历的年数减 1.

c 为该年的元旦算到某日为止的天数.

然后，用 7 除 s 的余数，即得星期几.

3.10 计算在绝对温度 T，波长为 λ 情况下黑体的辐射功率

$$E=\frac{2\pi ch\lambda^{-5}}{e^{ch/B\lambda T}-1}$$

其中 $c=2.997924\times10^8$ 是光速；$h=6.6252\times10^{-34}$ 是普朗特常数；$B=5.6687\times10^{-8}$ 为波尔兹曼常数. 编写一个程序键入 T 和 λ 的值，计算并输出 E 的值.

3.11 试编写一个程序，在屏幕上显示出如下表头和你本人的实际情况(这里不要求输出汉字).

NO. (学号)	Name (姓名)	Sex (性别)	Age (年龄)	Address (住址)	Total (入学总分S)

3.12 试编写一个程序，在打印纸上打出 3.2 题的结果.

3.13 编写一个程序，它可键入一个含四个字母的单词，并把组成该单词的字母按逆序输出.

3.14 编写一个程序，键入在 c 和 x 之间的任一字母，能打印出该字母在中间的相邻的五个大写字母. 例如，键入字母 F，输出

Five letters centred on F are DEFGH

3.15 编写一个程序，打印出如下三角形：

```
        *
      *   *
    *       *
  *           *
* * * * * * * * *
```

第 4 章　选择程序设计

用简单顺序结构只能表示很少的问题，大多数实际问题的算法都包含了选择结构或重复结构. 例如，求一元二次方程

$$ax^2+bx+c=0$$

的根. 程序 ex03_8 是在限定方程存在两个实根的条件下编写的. 其实，那个程序就不大实用，因为在运行程序之前，要手算一下判别式(b^2-4ac)，否则就可能发生运行中断. 一个实用程序应该使用方便，结果可靠，只要用户按规定要求输入，就应得出满意的结果. 要做到这一点，上述问题的算法将是一个多重选择结构，如图 4-7 所示.

表示选择结构的语句序列，就叫选择程序段. 这一章主要讲述包含选择程序段的程序设计. 为了描述选择结构，True BASIC 引入了结构 IF 语句、结构 CASE 语句等，因而，在 True BASIC 程序中，完全可以不用 GOTO 语句和行号，编写出完全结构化的程序，使得程序结构非常清晰.

4.1　逻辑表达式

选择结构也叫判断结构，为描述这个结构首先要学会描述判断条件. 在 True BASIC 语言中，判断条件是用逻辑表达式表示的.

True BASIC 语言没有规定完整的逻辑类型数据，也没有规定逻辑常数和变量的表示方法. 因此逻辑值不能存储、输入或输出. 逻辑表达式是由关系表达式和逻辑运算符组成的. 逻辑表达式可以用在 DO 循环和 IF 语句中.

4.1.1　关系表达式

(1) 关系运算符:

运算符	含　义
<	小于
<=或=<	小于等于
>	大于
=	等于
>=或=>	大于等于
<>或><	不等

关系运算符只能作同类型数据比较，而不能用于一个数值和一个字符串比较．数值比较和数学上的规定完全一致．字符串比较是采用字典方式．字符顺序是以 ASSII 代码为序的（见附录B)．注意，这种顺序大写字母排在小写字母之前，空格代码最小．

(2) 关系表达式的一般形式：

expr1 〈关系运算符〉 expr2

计算关系表达式，首先计算 expr 1 和 expr 2，然后再比较，判断是否成立．成立为真(True)，不成立为假(False)．

例如：		
	1+2<5	True
	3*5=4*0.5	False
	"a"<"b"	True
	"hi"[1:1]>"FOS"	True
	"a"& name $<"B"	False

4.1.2 逻辑表达式

(1) 逻辑运算符．

True BASIC 引入的逻辑运算符有三种：

not(非)　　and(与)　　or(或)

逻辑运算规则：设 r1 和 r2 为关系表达式，则

① r1 and r2 表示，当且仅当 r1 和 r2 同时为真时才为真，否则为假．

② r1 or r2 表示，当且仅当 r1 和 r2 同时为假时才为假，否则为真．

③ not r1 表示，若 r1 为真，则 not r1 为假，反之亦然．

上述逻辑运算规则如表 4-1 所示．

表 4-1 逻辑运算规则

r1	r2	not r1	r1 and r2	r1 or r2
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	Fale	True	False	False

例如：

x≤5 或 x>10，可以写作(x<=5)or(x>10)，

5<x≤10，可以写作(x>5)and(x<=10)．

注意，不准写作 5<x<=10．因为 5<x 为逻辑值，再用它和 10 比较没有意义．

(2) 逻辑表达式的一般形式．

逻辑表达式由关系表达式和逻辑运算符组成. 一个逻辑表达式可以包含算术运算、字符串运算、关系运算符和逻辑运算符. 逻辑表达式的运算顺序,一般地说和数学上的规定一致.

运算顺序如下:

a) 计算算术表达式的值(或字符串计算).

b) 计算关系表达式的值.

c) 计算逻辑表达式的值. 逻辑运算符的运算顺序是

not——>and——>or

d) 逻辑表达式若有多层括号,先作内层运算,后作外层运算.

e) 同层内优先次序相同的逻辑运算,不一定按自左向右顺序进行.

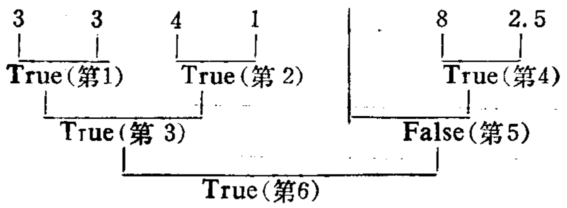
逻辑表达式运算顺序,如表 4-2 所示.

表 4-2 逻辑表达式运算符优先级规定

类 别	运 算 符	优先级
括号	( )	1
算术运算符	∧	2
	* /	3
	+ -	4
关系符	>,>=,=,<>,<,<=	5
逻辑运算符	not	6
	and	7
	or	8

例如,设 a=3,b=4,c=5. 那么,如下表达式的逻辑运算顺序是第 1 到第 6.

3*(c-b)=a and b > 1 or not 8*(b-a)>c/2



在书写逻辑表达式时,为了提高可靠性和易读性,建议多用圆括号,比如上式可写成:

(3*(c-b)=a) and (b>1) or not ((8*(b-a))>(c/2))

其实,True BASIC 当得到真值时,马上"化简"逻辑表达式. 换句话说,一旦确定了结果,表达式其余部分将不再去计算. 例如上述表达式,并没有计算 not8*(b-a)>c/2, 因为计算到第 3 步为真,则表达式总是真了. 这种验证方法是很巧妙的,效率大大提高.

4.1.3 逻辑函数

由逻辑函数得到逻辑值(True 或 False). True BASIC 给出 5 个逻辑函数,用于检查文件或 DATA 的数据状态.

KEY INPUT　当从键盘上读入最后一个数据后，若按了一个键则为真.

END DATA　若无数据项等待 READ 语句读时则为真，否则为假.

MORE DATA　若有数据项等待 READ 语句读时则为真，否则为假.

END #expr　若文件指针已到文件#expr 的结尾时则为真，否则为假.

MORE #expr　若文件指针未到文件#expr 的结尾时则为真，否则为假.

这 5 个函数，分别将在 5.2 节和第 9 章举例详述.

4.2 IF 语　句

IF 语句分为简单 IF 语句和结构 IF 语句，可以用来描述基本选择结构以及多重嵌套结构.

4.2.1 简单 IF 语句

(1) 简单 IF 语句形式:

IF 逻辑表达式 then 内嵌语句

IF 逻辑表达式 then 内嵌语句 1 else 内嵌语句 2

其中内嵌语句是一个可执行的简单语句，但不能是简单 IF 语句. 为了和基本 BASIC 兼容，也可以是行号，然而并不提倡使用.

(2) 应用举例.

例 1　打印出某一变量的绝对值.

解　这里不用标准函数 abs(x)，而是直接用如下公式计算:

$$x=\begin{cases} x, & x\geq 0 \\ -x, & x<0 \end{cases}$$

算法 N-S 图　如图 4-1 所示.

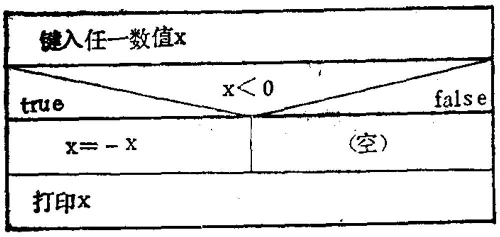


图 4-1

根据图 4-1，可编程序如下:

```
PROGRAM ex04_1
INPUT prompt "x="; x
IF x < 0 then LET x=-x
PRINT "x=";x
END
```

例 2　编写计算 y 值的程序，公式如下:

$$y=\begin{cases} \dfrac{\sin x}{x}, & x\neq 0 \\ 1\ , & x=0 \end{cases}$$

解　算法 N-S 图如图 4-2 所示.

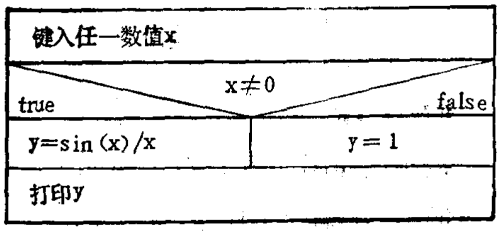


图 4-2

可编程序如下:

```
PROGRAM ex04_2
INPUT prompt "x=":x
```

```
IF x <> 0 then LET y=sin(x)/x else LET y=1
PRINT "y=";y
END
```

4.2.2 结构 IF 语句

简单 IF 语句用于描述各分支只有一个操作的选择结构比较方便. 为了描述各分支有一个或更多个操作的选择结构,True BASIC 引入了结构 IF 语句,它有三种形式.

(1) 结构 IF 语句的形式之一:

```
IF 逻辑表达式 then
    ……
    ……  } 语句块
    ……
END  IF
```

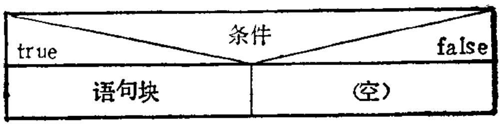


图 4-3

N-S 图如图 4-3 所示.

例如,任意输入两个数,赋给变量 a,b,试编一个程序,使 a≥b,即把 a, b 按降序排列.

其算法 N-S 图如图 4-4 所示.

依据图 4-4 可编程序如下:

```
PROGRAM ex04_3
INPUT prompt "a,b?":a,b
IF a < b then
   LET wk=a
   LET a=b
   LET b=wk
END IF
PRINT "a=";a,"b=";b
END
```

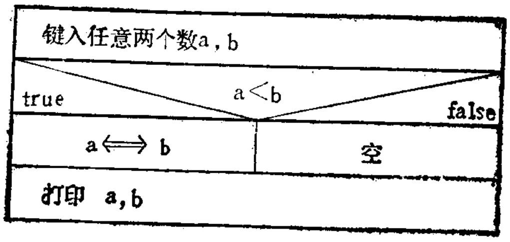


图 4-4

(2) 结构 IF 语句形式之二:

```
IF 逻辑表达式 then
    ……
    ……  }     语句块 1
    ……
ELSE
    ……
    ……  }     语句块 2
    ……
END IF
```

控制流程图如图 1-7(b) 所示; N-S 图如图 1-9(b) 所示.

例 1 试编一个程序,计算某工厂工人每周的工资. 具体方法如下:

① 每周工作未超过 40h(小时)者,按 4.5 元/h 计算,交所得税 0.18%.

② 每周工作超过 40h 者,其中 40h 按 4.5 元/h 计算, 超过部分按 6.75 元/h 计算, 交所得

税 0.2%.

解

方案 A $\begin{cases} gross = 4.5 \times hours \\ tax = 0.18\% \times gross \end{cases}$ $hours \leqslant 40$

方案 B $\begin{cases} gross = 180 + 6.75(hours - 40) \\ tax = 0.2\% \times gross \end{cases}$ $hours > 40$

计算工资的 N-S 图如图 4-5 所示.

按照图4-5,可编程序如下:

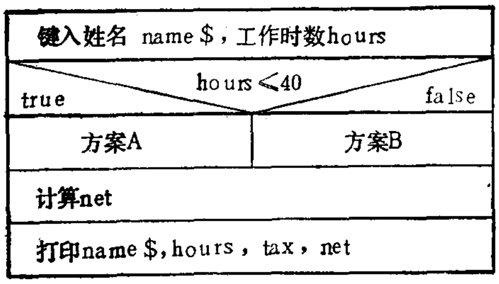


图 4-5

```
PROGRAM ex04_4
INPUT prompt "Name$,hours?":name$,hours
IF hours<=40 then
    LET gross=4.5*hours
    LET  tax=0.18*gross/100
ELSE
    LET gross=180+6.75*(hours-40)
    LET tax=0.20*gross/100
END IF
LET net=gross-tax
PRINT name$;tab(15);"hours=";hours
PRINT "Tax=";tax;tab(15);"Net=";net
END
```

运行结果:

```
Name$, hours? lu jian,30
lu jian        hours=30
Tax= .243      Net=134.757
```

例 2 编写一个猜字游戏程序.

提示:一枚硬币丢下去,猜哪一面向上称为猜字游戏.

解 这里要用随机函数 rnd 生成 0 和 1 摸拟两种状态. 具体办法是用函数 round(rnd)来实现. 为了使随机函数 rnd 具有较好的随机性, 这里使用了 RANDOMIZE 语句. 程序如下:

```
PROGRAM ex04_5
! a simple guessing game
!
RANDOMIZE
LET answer=round(rnd)
INPUT prompt "your guess?": guess
IF guess=answer then
    PRINT "right."
ELSE
    PRINT "wrong."
```

```
        PRINT "It was";answer
    END IF
    END
```

(3) 结构 IF 语句形式之三:

IF 逻辑表达式 then

……
……} 语句块 1
……

ELSE if 逻辑表达式 then

……
……} 语句块 2
……

ELSE

……
……} 语句块 3
……

END IF

其中 ELSE if 子句,可以有多个.

这是基本选择结构的多重嵌套,如图 4-6.

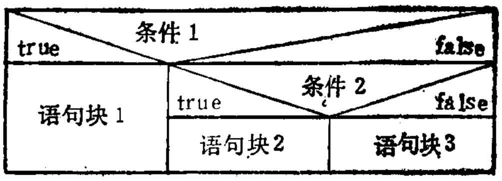


图 4-6

例 求下列一元二次方程的根:

$$ax^2+bx+c=0$$

解 任意给定 a,b,c,可能会出现如下情况:

① $a=0, b=0$,方程无意义.

② $a=0, b\neq0$,方程有一个根.

③ $a\neq0, b\neq0, b^2-4ac\geqslant0$,有两个实根.

④ $a\neq0, b\neq0, b^2-4ac<0$,有两个复根.

根据上述分析,可画 N-S 图如图 4-7 所示.

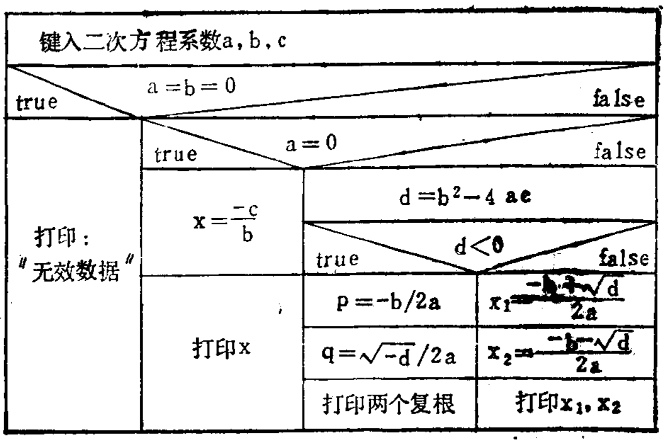


图 4-7

根据图 4-7 可编程序如下:

```
PROGRAM ex04_6
!    roots of ax^2+bx+c=0
!
INPUT prompt "a,b,c?":a,b,c
IF (a=0)and(b=0)then
  PRINT "illigal data."
ELSE if a=0 then
  PRINT "x=";-c/b
ELSE
  LET d=b*b-4*a*c
  IF d<0 then
```

```
      LET p=-b/(2*a)
      LET q=sqr(-d)/(2*a)
      PRINT "x1=";p; "+";q;"i",
      PRINT "x2=";p;"-";q;"i"
   ELSE
      PRINT "x1=";(-b+sqr(d))/(2*a),
      PRINT "x2=";(-b-sqr(d))/(2*a)
   END IF
END IF
END
```

4.3 CASE 语 句

IF 语句是在两种情况中作出选择，而 CASE 语句则是在多种情况中作出选择. 它是多重 IF 结构的特殊形式.

4.3.1 SELECT CASE 语句(选择情况语句)的一般形式

```
SELECT CASE expr
    CASE vlist1
        ......
        ......  } 语句块 1
        ......
    CASE vlist2
        ......
        ......  } 语句块 2
        ......
    ...
    CASE ELSE
        ......
        ......  } 语句块 n
        ......
END SELECT
```

其中:expr 是字符串表达式或数值表达式,不能是逻辑表达式.

vlist 是 value1, value2, …

value 和 expr 类型一致. 有如下三种形式:

常数 (表示某个值,如 10)

常数 1 to 常数 2 (表示值域,如"A"to"D")

is 关系操作符 常数 (表示值域,如 is<100)

CASE 语句的执行过程如图 1-13 所示. 首先计算表达式 expr 的值,然后从上至下,从左至

右,依次查找与 expr 值匹配的值或值域.如果成功,则执行相应的语句块,否则执行 ELSE 后的语句块,最后转至 END SELECT 语句后的语句.

注意:SELECT CASE 的 value,必须是互不相同的,否则会出现二义性,使程序无法执行.

4.3.2 应用举例

例 1 键入一个数 n(1≤n≤7),打印出星期 n 的英文单词.

解 用 CASE 语句可编程序如下:

```
PROGRAM ex04_7
!
INPUT prompt " 1<= n <= 7: ";n
SELECT CASE n
  CASE 1
          PRINT "Monday"
  CASE 2
          PRINT "Tuesday"
  CASE 3
          PRINT "Wedneday"
  CASE 4
          PRINT "Thursday"
  CASE 5
          PRINT "Friday"
  CASE 6
          PRINT "Saturday"
  CASE 7
          PRINT "sunday"
  CASE else
          PRINT "illegling data."
  END SELECT
END
```

例 2 菜单功能调用程序.

解 用 CASE语句可编程序如下:

```
PROGRAM ex04_8
PRINT"Dos function Menu"
PRINT "Edit", "Compile","Link", "Run", "END"
INPUT prompt "Piase input upper letter:":string$
SELECT CASE string$
  CASE "E"
       CALL Edit
  CASE "C"
       CALL Compile
```

```
CASE'L'
    CALL link
CASE 'R'
    CALL Run
CASE 'END'
CASE else
    INPUT prompt 'input again:':string$
END SELECT
END
```

这个程序是控制程序，通过它的执行调用某个子程序. 当键入 END 时，执行空块后，转到程序结束语句.

习 题

4.1 编写一个按下述公式计算 y 值的程序(x 值由 INPUT 语句输入)

$$y=\begin{cases}\dfrac{\sin(x)+\cos(x)}{2}, & x\geq 0\\[2mm] \dfrac{\sin(x)-\cos(x)}{2}, & x<0\end{cases}$$

4.2 键入 a,b,c 三个数，编一个程序，它能把这三个数按其值的大小顺序打印出来.

4.3 编写一个程序，用它确定两个都只有三个字母组成的单词在辞典中的先后顺序. 例如：输入

the, are↙

输出

the > are

4.4 编写一个程序，若键入任意三个大于零的数，则可确定以这三个数为边长是否能构成一个三角形. 若能，再确定它是否是一个直角三角形.

4.5 键入任一字母，然后把它输出，若该字母在字母表的前半部时，则输出其后续字母；若它在后半部时，则输出它之前的字母.

4.6 编写一个程序，能按如下形式键入年、月、日：

××/××/××　　(日/月/年)

然后输出日期的完整形式. 例如：输入

23/ 4/88

输出

23rd　　Apr　　1988

注意：1 月至 12 月相继表示为 Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec. 日子的末位为 1 附上 st，末位为 2 附上 nd，末位为 3 附上 rd，其它附以 th.

4.7 利用 case 语句，把下面的语句改为结构更清晰的语句：

```
if ch$="<" then
   LET b$="x<y"
else if ch$=">" then
   LET b$="x>y"
else if ch$="=" then
```

```
        LET b$='x=y'
    else
      LET    error$='true'
    end if
```

4.8 设w为托运货物重量(单位: kg),试按如下公式计算托运费:

$$s=\begin{cases}0\ , & w\leqslant 20\\ 0.15(w-20), & 20<w\leqslant 40\\ 0.25(w-40)+3, & 40<w\leqslant 60\end{cases}$$

若超过60kg,则不允许随身携带上车,不予计费(要求用CASE语句编写程序).

4.9 美国某大学收学费的办法是,本州学生听课12门以下交350元,超过12门后,每多听一门多收20元.非本州学生听课12门以下交800元,超过12门后,每多听一门加收45元.试编一个计算学费的程序.

4.10 有8个小球,其中7个重量相同,有一个较轻,试用天平称两次,找出其中较轻的一个.

[提示] 试编一个程序,以8个变量表示8个小球,任意送入7个相同的数和一个较小的数,打印出较小的数.

4.11 某工厂的工人工资卡中载有以下项目:职工号、基本工资率(元/h),本周班次(分1,2,3班)和实作工时(h).

(1) 工资计算公式:

毛收入=(基本工资率×工资系数)×实作工时

班　次	1(白班)	2(中班)	3(晚班)
工资系数	1	1.25	1.50

(2) 纳税标准

每周毛收入(元)	100.0以下	100.0~149.99	150.0~199.99	200.0~299.99	300.0以上
税　率	0	8%	12%	15%	7%

试编一个程序,读入上述工资卡的各项之后,能计算出工人的毛收入,应纳税款与一周的净收入,并随同职工号一起打印出来.

第5章 循环程序设计

循环结构如图1-7(c)所示. 表示循环结构的语句序列称为循环程序段. 这一章主要讲述包含循环程序段的程序设计. True BASIC引入了功能极强的循环结构语句,包括DO-LOOP语句,条件DO-LOOP语句和FOR-NEXT语句.

5.1 循环概述

循环结构也叫重复结构. 虽然,每次循环执行的语句是相同的,而操作的对象(数据)经常是不同的. 通常循环可分类如下:

循环 { 无限循环
　　 { 有限循环 { 条件型循环
　　　　　　　 { 计数型循环

所谓无限循环,指的是循环次数是无限的,即可无休止地运行下去.

(1) 无限循环结构语句形式:

DO

……
…… } 循环体(语句块)
……

LOOP

执行它,要重复执行语句块. 在IBM PC机上,可按STOP键(即[Ctrl]+[Scroll Look])中断程序运行.

(2) 应用举例.

例1 求若干数的和(也叫累加和).

解 与计数法类似,基本算法是

sum=0

sum=sum+xi　　(i=1,2,…)

可编程序如下:

```
PROGRAM ex05_1
! use the stop key to make this program stop
!
LET sum=0
DO
```

```
      INPUT prompt "x?":x
      LET sum=sum+x
      PRINT "sum=";sum
LOOP
END
```

无限循环不符合结构化程序设计准则，所以，除特殊情况外，一般不要编这样的程序.

例 2 试编写一个十字路口交通控制程序.

解 十字路口东西(east and west)两路口一样，有红(red)、黄(yellow)、绿(green)三个灯，南北(south and north)两个路口一样，有红、黄、绿三个灯，而且是交互变换的. 这里用 6 个字符串变量表示 6 个灯.

可编程序如下:

```
PROGRAM ex05_2
!
DO
     LET ewr$, ewy$, sny$, sng$="back"
     LET snr$="red"
     LET ewg$="green"
     PRINT "EW";ewg$,"SN":snr$
     PAUSE 5
     LET ewy$,sny$="yellow"
     PRINT "EW"; ewy$,"SN";sny$
     PAUSE 1
     LET snr$,sny$,ewg$,ewy$="back"
     LET ewr$="red"
     LET sng$="green"
     PRINT "EW";ewr$,"SN";sng$
     PAUSE 5
     LET ewy$,sny$="yellow"
     PRINT "EW";ewy$,"SN";sny$
     PAUSE 1
LOOP
END
```

运行结果:

```
EW  green            SN  red
EW  yellow           SN  yellow
    EW  red              SN  green
    EW  yellow           SN  yellow
EW  green            SN  red
EW  yellow           SN  yellow
    EW  red              SN  green
```

```
    EW  yellow          SN  yellow
EW  green             SN  red
EW  yellow            SN  yellow
    EW  red             SN  green
    EW  yellow          SN  yellow
EW  green             SN  red
EW  yellow            SN  yellow
    EW  red             SN  green
    EW  yellow          SN  yellow
EW  green             SN  red
    program stopped.
```

说明:

PAUSE 语句是暂停语句. 一般形式为

PAUSE expr

其中 expr 为算术表达式.

执行它,程序运行暂停 expr 秒. expr 的值,可以是几秒,十分之一秒,百分之一秒等任意值.

5.2 条件型循环

它的特点是:在程序运行前,不知道循环次数,但是在每次循环中,至少有一个与循环控制条件有关的变量的值要改变,当它使得预定的循环结束条件满足时,则循环结束.

5.2.1 当型循环

它的结构如图 1-9(c)所示. 为描述它,True BASIC 引入了如下结构语句:

DO while 逻辑表达式

```
......)
......}循环体(语句块)
......)
```

LOOP

执行它,当(while)逻辑表达式的值为真时,则重复执行循环体, 当逻辑表达式的值为假时,则循环结束,转去执行 LOOP 之后的可执行语句.

例 求累加和,当输入数为 0 时结束.

解 根据题意不难画出 N-S 图,如图 5-1 所示.

根据图 5-1 可编程序如下:

```
PROGRAM ex05_3
! Input conditional loop
!
LET sum=0
INPUT prompt "x?":x
```

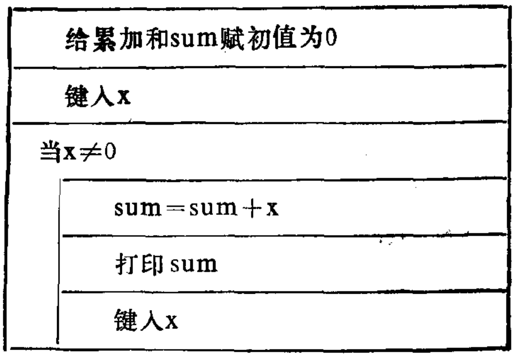


图 5-1

```
DO while x < > 0
     LET sum=sum+x
     PRINT "sum=";sum
     INPUT prompt "x=":x
LOOP
END
```

5.2.2 直至型循环

如图 1-10 所示,这是先处理后判断的循环. 循环体至少要执行一次. 为便于描述它, True BASIC 引入了如下结构语句.

DO

……
……}循环体(语句块)
……

LOOP until 逻辑表达式

执行它,先作循环体再判断. 如果逻辑表达式为假,则继续执行循环体;如果逻辑表达式为真,则循环结束,转去执行 LOOP 之后的可执行语句.

例 求累加和,当和大于或等于 10000 时结束.

解 N-S 图 如图 5-2 所示.

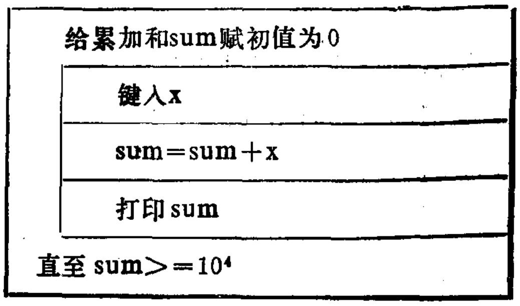


图 5-2

根据图 5-2 可编程序如下:

```
PROGRAM ex05_4
! conditional loop
```

```
LET sum=0
DO
    INPUT prompt "x=";x
    LET sum=sum+x
    PRINT "sum=";sum
LOOP until sum >=1e4
END
```

5.2.3 带 EXIT 的无限循环

如图 1-11 所示,这也是常见的循环结构. 为便于描述它, True BASIC 引入了如下结构语句,就是在 DO-LOOP 循环之中包含一个出口语句,形式如下:

```
DO
    ......
    ......}循环体(包含 EXIT DO 语句)
    ......
LOOP
```

执行 EXIT DO 语句, 控制直接转至 LOOP 之后的第 1 条可执行语句. 通常 EXIT DO 语句是在条件控制语句(如, IF 语句、块 IF 或 CASE 语句)之内,否则循环结构将是无意义的.

例 求累加和,当输入数为 0 时结束.

解 用 EXIT DO 语句,N-S 图如图 5-3 所示.

依据图 5-3,可编程序如下,它比 ex05-3 简单清晰.

```
PROGRAM ex05_5
! Input conditional loop
!
LET sum=0
DO
    INPUT prompt "x?":x
    IF x=0 then exit do
    LET sum=sum+x
    PRINT "sum=";sum
LOOP
END
```

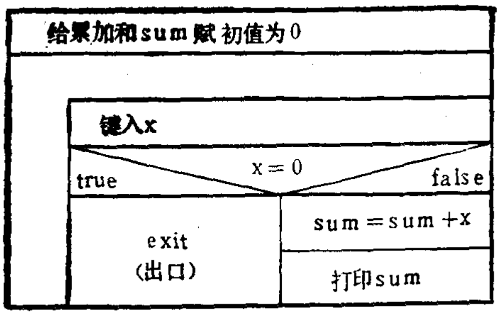


图 5-3

5.2.4 带 EXIT 的当型循环或直至型循环

这两种循环结构也是扩展的循环结构,可用基本结构复合表示,这里不一一详述. 它们也是常见的结构,为此 True BASIC 引入了如下语句:

```
DO while 逻辑表达式
    ......
    ......}循环体(包含 EXIT DO 语句)
    ......
LOOP
```

和

DO

```
……
……  循环体(包含 EXIT DO 语句)
……
```

LOOP until 逻辑表达式

这两个都是双重判断的循环结构.

例 求累加和,当总和≥10000 或输入数据为 0 时,则求和结束

解 N-S 图如图 5-4 所示.

依据图 5-4 可编程序如下:

```
PROGRAM ex05_6
! Input conditional loop
!
LET sum=0
DO
    INPUT prompt "x?":x
    IF x=0 then exit do
    LET sum=sum+x
    PRINT "sum=";sum
LOOP until sum >=1e4
END
```

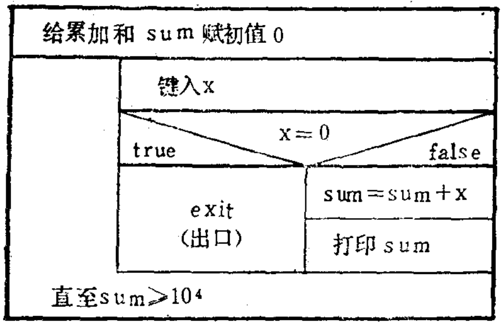


图 5-4

执行这个程序,当输入数据 x 为 0 时,执行 exit do 语句可直接转至 END 语句结束. 也可在和 sum 大于或等于 10^4 时,循环正常结束,转去执行 END 语句结束.

5.2.5 条件型循环应用举例

例 1 假定在银行存款 10000 元,银行月利率 0.85%,问几个月后存款增加一倍?

解 算法分析:

1 个月后本利=本钱+本钱*利率=本钱(1+利率)

2 个月后本利=1 个月后本利*(1+利率)

……

n 个月后本利=n−1 月后本利*(1+利率)

根据上述分析,可编程如下:

```
PROGRAM ex05_7
! stop when the money has doubled.
!
LET old=10000
LET new=old
LET interest=8.5/1000
DO until new >=2*old
    PRINT month,new
```

```
        LET month=month+1
        LET new=new*(1+interest)
LOOP
PRINT "In";month;" month, You'll have";new
END
```

例 2 用牛顿迭代公式计算变量 a 的平方根(开方).

解 设任意变量 a 的平方根为 x,则

$$x=\sqrt{a}$$

即

$$f(x)\equiv x^2-a=0.$$

事实上,开方计算就是求一个方程的根. 牛顿迭代法的一般公式为

$$x_{n+1}=x_n-\frac{f(x)}{f'(x)}$$

不难导出计算 a 平方根的递推公式:

$$x_{n+1}=\frac{1}{2}\left(x_n+\frac{a}{x_n}\right)\qquad(n=1,2,3,\cdots)$$

任意给定一个初值 x_1,用上述公式,可以计算出一个数列:

$$x_1,x_2,x_3,x_4,\cdots,x_n,x_{n+1}\cdots$$

牛顿迭代法告诉我们,当 n 逐步增大时,若$|x_{n+1}-x_n|$趋近于 0,则 x_{n+1} 逐步逼近$\sqrt{a}$的值. 本题要求$|n_{n+1}-x_n|<10^{-5}$时结束.

若把数列看作变量 x 不同时刻的值,则可直接写出迭代公式:

$$x=a,\ x=\frac{1}{2}\left(x+\frac{a}{x}\right)$$

通过这个例题可以看出,在科学计算中,只要存在递推公式,就可化作迭代公式. 在编程序时就可用条件循环结构来描述,如图 5-5 所示.

该题可编程序如下:

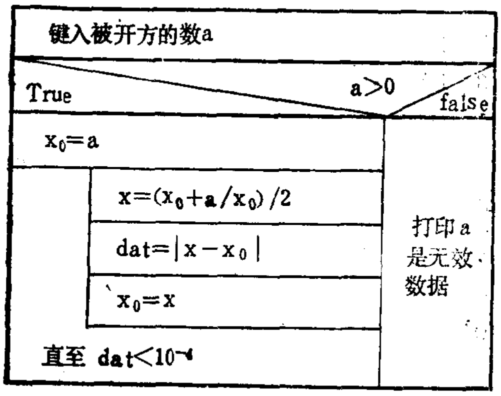


图 5-5

```
PROGRAM ex05_8
!      Newton method
INPUT prompt "a=":a
IF a > 0 then
   LET xold=a
   DO
      LET x=(xold+a/xold)/2
      LET dat=abs(x-xold)
      LET xold=x
   LOOP until dat < 1e-5
   PRINT "a=";a,"x=";x
ELSE
   PRINT a;" is illigal data."
```

```
END IF
END
```

例 3 统计某班学生期末考试平均成绩的分布情况. 设平均成绩为 p，分数段规定如下：$100\geq p\geq 90$，优秀；$90>p\geq 80$，良好；$80>p\geq 70$，中等；$70>p\geq 60$，及格；$p<60$，不及格，并求学生总数.

解 统计学生成绩分布情况，包含了多个分支求累加和的运算. 由于学生人数未知，所以要设定循环结束标志. 这里取 $p>100$ 或 $p<0$ 为结束标志，即可用条件循环结构嵌套一个多支选择结构来描述，如图 5-6 所示.

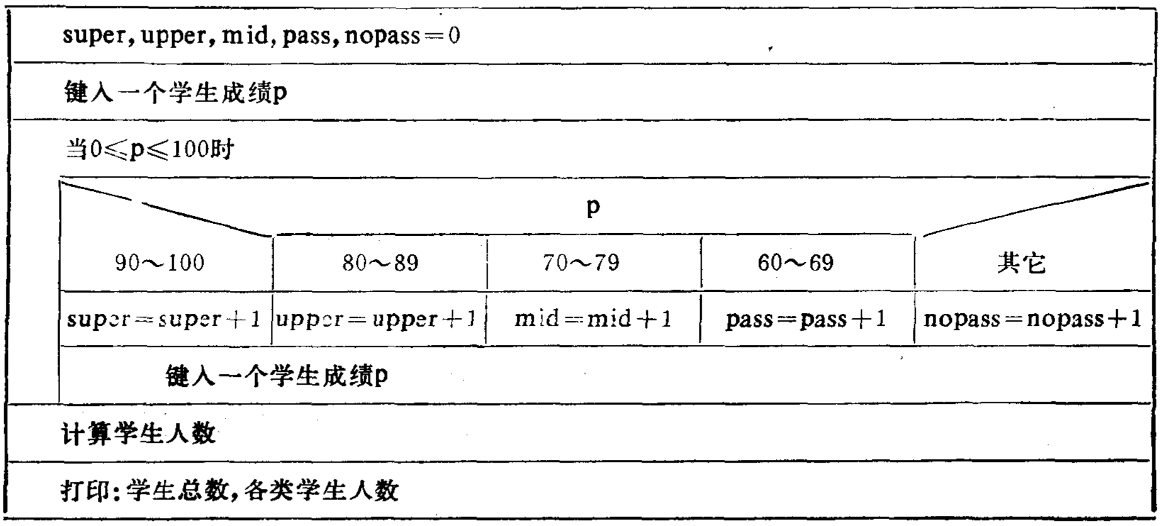


图 5-6

这里各等级人数分别记作 super(优秀)，upper(良好)，mid(中等)，pass(及格)，nopass(不及格).

可编程序如下：

```
PROGRAM ex05_9
!       statistical result of students
LET super, upper, mid, pass, nopass=0
INPUT prompt "p=":p
DO while (p<=100) and (p>=0)
    SELECT CASE p
    CASE 90 to 100
          LET super=super+1
    CASE is >=80
          LET upper=upper+1
    CASE is >=70
          LET mid=mid+1
    CASE is >=60
          LET pass=pass+1
```

```
    CASE else
        LET nopass=nopass+1
    END SELECT
    INPUT prompt "p=":p
LOOP
LET n=super+upper+mid+pass+nopass
PRINT "n=";n
PRINT "super=";super, "upper=";upper,
PRINT "mid=";mid, "pass=";pass, "nopass=";nopass
END
```

例4 时钟程序.

解 利用三个记数器，分别模拟时钟的时、分、秒. 为实现60秒进位到分，60分进位到时，达到24小时要清零，这里要用嵌套的CASE语句. 该程序运行速度可用PAUSE语句调整. 停止运行要用STOP键.

具体程序如下:

```
PROGRAM ex05_10
! clock
INPUT prompt "hour?":hour
INPUT prompt "minute?":minute
LET minute0=minute
INPUT prompt "second?":second
DO
    SELECT CASE second
    CASE 0 to 58
            LET second=second+1
            PAUSE 0.1
    CASE 59
            LET second=0
            SELECT CASE minute
            CASE 0 to 58
                    LET minute=minute+1
            CASE 59
                    LET minute=0
                    SELECT CASE hour
                    CASE 0 to 22
                            LET hour=hour+1
                    CASE 23
                            LET hour=0
                    CASE else
                    END SELECT
```

```
            CASE else
            END SELECT
        CASE else
        END SELECT
        PRINT using "##:##:##":hour,minute,second
    LOOP
    END
```

5.3 计数型循环

计数型循环的特点是，在程序运行前，知道(或可以计算出)循环次数. 循环由一个控制变量(或叫计数器)计数，当循环控制变量的值超过预先给定的界限时，则循环结束.

5.3.1 计数型循环结构语句形式:

```
FOR var=first TO last STEP size
    ......)
    ......}循环体
    ......)
NEXT var
```

其中 var 为数值型变量，叫循环控制变量. first, last, size 为数值表达式，分别称为初值、终值和步长.

FOR-NEXT 循环执行过程，如图 5-7 所示.

执行 FOR-NEXT 循环，相当于执行如下程序段:

```
LET m1=fist
LET m2=last
LET m3=size
LET n=max(int((m2-m1+m3)/m3),0)
IF n>0 then
  LET var=m1
  IF m3>0 then
     DO while var<=m2
          循环体
        LET var=var+m3
     LOOP
  ELSE
     DO while var>=m2
          循环体
        LET var=var+m3
     LOOP
```

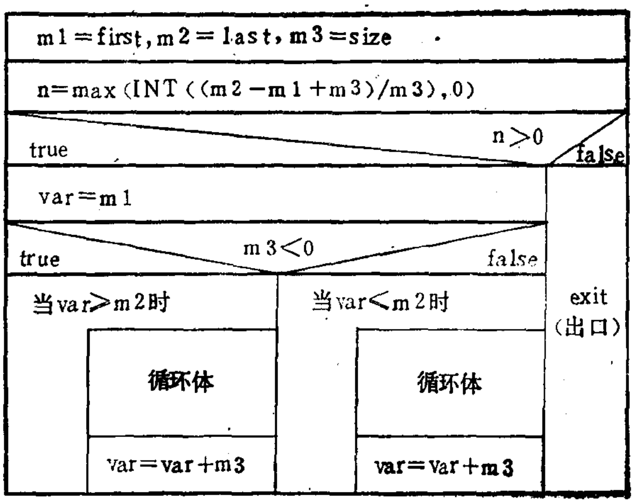


图 5-7

```
    END IF
END IF
```

从这里可以看出：FOR-NEXT 循环结构语句是基本循环结构的扩展，引入它使得程序更加结构化了。本书把计数型循环选作基本结构，并规定它的 N-S 图如图 5-8 所示。

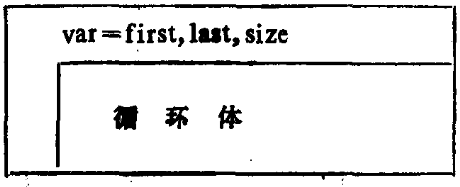


图 5-8 计数型循环 N-S 图

5.3.2 计数型循环的几点说明

(1) 当 size=1 时，step 部分可以省略；当 size=0 时，为无限循环。

(2) 先判断，后执行。这样，循环体可能一次也不被执行。

(3) 控制变量 var 在循环体内可以被引用；当控制转出循环后，在循环体外，var 的值也可以被引用。

(4) 控制变量在循环体内可以被赋值。这一点要特别注意：当控制变量被赋值后，循环次数随之改变。

(5) EXIT FOR 语句，可以用于 FOR-NEXT 循环体之中，构成双重控制。执行它，可以从循环中转出去执行 NEXT 后第一个可执行语句。

例 试编一个程序，打印出不大于 n 的所有偶数。

解 这个问题有多种编程方法：

程序 1：

```
PROGRAM ex05_11a
INPUT prompt"n=":n
FOR k=1 to n
     LET m=k*2
     IF m>n THEN EXIT FOR
     PRINT m;
NEXT k
PRINT
END
```

运行结果：

```
n=10↙
  2   4   6   8   10
```

程序 2：

```
PROGRAM ex05_11b
INPUT prompt"n=":n
FOR k=1 to n
```

```
        LET k=k+1
        PRINT k;
  NEXT k
  PRINT
  END
```

运行结果:

```
n=10↲
   2   4   6   8   10
```

在 ex05_11b 中,因为控制变量 k 在循环体内再一次被赋值,所以,原来可循环 10 次,现在只循环 5 次,改变了循环次数.

程序 3:

```
PROGRAM ex05_11c
INPUT prompt "n=":n
FOR k=2 to n step 2
     PRINT k;
NEXT k
PRINT
END
```

这个程序利用了循环参数可以是任意数字表达式的规定,恰当地选择了初值和增量,所以程序简洁又准确.

5.3.3 计数型循环应用举例

常见的循环问题是:求累加和("减"化作加负数),求连乘积("除"化作乘倒数),以及求最大值、最小值等,这时对所求的变量(如和、积等)都要赋初值,一般方法是:

(1) 求累加和,赋初值为第 1 项或为 0.

(2) 求连乘积,赋初值为第 1 项或为 1.

(3) 求最大值,赋初值为第 1 项或足够小的量.

(4)求最小值,赋初值为第 1 项或足够大的量. 而且,赋初值语句要放在循环结构开头语句之前,切勿放在循环体内,否则徒劳无功.

例 求 1 到 12 的阶乘.

解 $n!=n*(n-1)*\cdots 2*1$.

按递推算法,即

$$n!=(n-1)!*n$$

可编程序如下:

```
PROGRAM ex05_12
! computing
PRINT " n"," n!"
LET farc=1
```

```
5  FOR n=1 to 12
6      LET farc=farc*n
7      PRINT n, farc
8  NEXT n
9  END
```

运行结果:

n	n!
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800
11	39916800
12	479001600

这里阶乘初值为 1,而且放在 FOR 语句之前.

例 2 求 e 的近似值. 要求不超过 10 项,或满足精度 10^{-6} 为止.

解 由数学分析知道:

$$e^x=1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots+\frac{x^n}{n!}+\cdots$$

令 $x=1$,则有

$$e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{n!}+\cdots$$

不难发现前后项之间有如下递推关系:

$$term_n=term_{n-1}/n$$

这样,可以得到如下迭代公式:

$$n=n+1$$
$$term=term/n$$
$$e=e+term$$

初值分别为

$$n=0$$
$$term=1$$
$$e=1$$

这个问题的算法是一个计数型循环,而又有条件控制, 所以是一个双重控制循环结构,可编程序如下:

```
PROGRAM ex05_13
LET e, term=1
FOR n=1 to 10
      LET term=term/n
      LET e=e+term
      IF term<1e-6 then EXIT FOR
NEXT n
PRINT"term=";term,"e=";e
END
```

这里使用了 EXIT FOR 语句，当精度被满足时，则转出循环. EXIT FOR 语句只能用在 FOR-NEXT 循环之中.

例 3 用蒙特卡逻法计算图 5-9 所示图形面积 S.

解 Monte Corlo 法具体步骤是，首先作一个矩形 $0\leqslant x\leqslant\pi, 0\leqslant y\leqslant 1$（矩形面积为 π）. 然后随机地对矩形域投入大量的点，若它们均匀分布在矩形区域内，则

$$\frac{\text{S 的面积}}{\text{矩形面积}}\approx\frac{\text{落在 S 中的点数}}{\text{总的点数}}$$

所以

$$S\approx\frac{\text{落在 S 中的点数}}{\text{总的点数}}\pi$$

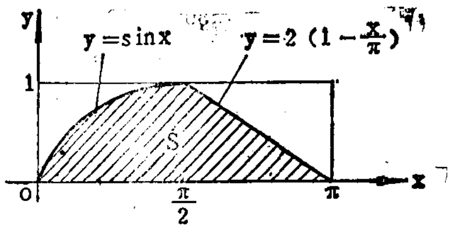


图 5-9 Monte Carlo 法

依据上述公式，首先要模拟矩形区域内随机地投点，其办法是：用均匀分布的随机函数 rnd 表示矩形区域内随机点的坐标 (x,y)，即

$$x=\text{rnd}\cdot\pi,\ \text{rnd}\in[0,1]$$
$$y=\text{rnd}$$

然后，分别统计投掷的‘总的点数’与‘落在 S 域中的点数’，最后代入上述计算 S 的公式，即可求得面积. 为了便于分析 Monte Carlo 法结果，还计算了精确解和相对误差：

$$so=1.0+0.25\pi$$
$$data=(so-s)/so$$

依照上述算法，可编写程序如下：

```
PROGRAM ex05_14
! Monte Corlo method(m>200)
OPEN #5:printer
RANDOMIZE
INPUT prompt"m?":m
PRINT"I","N","S","DATA"
LET n=0
FOR i=1 to m
      LET x=rnd*pi
```

```
        LET y=rnd
        IF x>=0.5*pi then
           LET y1=2*(pi-x)/pi
           IF y<=y1 then LET n=n+1
        ELSE
           LET y1=sin(x)
           IF y<=y1 then LET n=n+1
        END IF
        IF mod(i,200)=0 then
           LET s=n/i*pi
           LET so=1+0.25*pi
           LET data=(so-s)/so
           PRINT i,n,s,data
        END IF
NEXT i
END
```

从理论上说，只要投点数量足够多，结果应该是精确的，下面是投点1000次，每间隔 200 次输出一组数据，结果如下：

m?1000

I	N	S	DATA
200	126	1.9792	-.10855
400	239	1.8771	-.051363
600	358	1.87448	-4.98967e-2
800	469	1.84176	-3.15675e-2
1000	573	1.80013	-8.25274e-3

5.4 READ语句与DATA语句

前面已讲述了 LET 和 INPUT 语句给变量赋值的方法．READ 和 DATA 语句是另一种赋值的方法．

5.4.1 READ 和 DATA语句一般形式

```
READ var11,var12,…
READ var21,var23,…
 ⋮
DATA item11,item12,…
DATA item21,item22…
```

其中 var 为变量，可以是数值型，也可以是字符串型，但不准许是数组变量，item为数据项，

是数值或字符串，它要与 var 在顺序、类型上匹配．这里数值也可看作是字符串．

READ 语句是可执行语句，根据编程需要书写编排，而 DATA 语句是不可执行语句，所有的DATA 语句合起来，提供了一个数据表，它可以集中或分散地放在程序的任何位置，通常是放在程序的尾部．True BASIC 将自动跳过一切不可执行语句和注释，按照程序控制执行．执行 READ 语句，则将 DATA 数据项中的数据一个接一个地读出，再赋给变量．形象地说，有一个变量指针和一个数据项指针，开始各自指向开头第 1 个，随着变量指针从头到尾移动，数据项指针也逐步向后移动．

例如:

```
PROGRAM ex05_15
! trivia quiz.
!
READ num_quest
FOR i=1 to num_quest
     READ question$, answer$
     PRINT question$;
     LINE INPUT reply$
     IF reply$=answer$ then
        LET right=right+1
        PRINT"correct"
     ELSE
        PRINT"No, the correct answer is";answer$;"."
     END IF
NEXT i
PRINT"You got";100*right/num_quest;"%right."
DATA 4
DATA what is the capital of Austria, Vienna
DATA what year did Franklin pierce take office, 1853
DATA "How many years, on average, does a baboon live", 20
DATA How about a gray squirrel, 5
END
```

注意:在这个程序中，READ 的变量个数和DATA 语句的数据个数都是 9 个．

5.4.2 DATA 数据项系统判断条件

在程序 ex05_15 中，READ 变量个数和 DATA 数据项个数正好相等．如果不等就会出现运行错误信息．这就要求在变更或书写数据项时，要特别小心．为了给程序员稍许宽松的环境，True BASIC 引入了如下 DATA 数据判断条件:

MORE DATA　表示如果 DATA 还有数据可供读入，判断条件为真．

END DATA　表示如果 DATA 的数据已读完，即无数据可供读入，判断条件为真．

若把这两个条件用在条件循环中，无疑更方便了．当然，这两个条件也可用在 IF 语句和结

构 IF 语句之中.

例如:

```
PROGRAM ex05_16
! trivia quiz.
!
DO
  READ question$, answer$
  PRINT question$;
  LINE INPUT reply$
  IF reply$=answer$ then
     LET right=right+1
     PRINT"correct"
  ELSE
     PRINT"No, the correct answre is"; answer$;"."
  END IF
LET num_quest=num_quest+1 !count questions
LOOP until end data
PRINT"You got";100*right/num_quest;"%right."
DATA What is the capital of Austria, Vienna
DATA What year did Franklin pierce take office, 1853
DATA"How many years, on average, does a baboon live", 20
DATA How about a gray squirrel, 5
END
```

这样修改后,若想在 DATA 语句中增加或删除数据项就方便了.

5.4.3 DATA 数据的重复使用

在程序 ex05_16 中,DATA 数据只使用了一次,为了能重复使用,True BASIC 引入了复位语句, 即

RESTORE

执行它,把数据项指针复位,指向数据表的第一项,即第 1 个 DATA 语句的第 1 项.

例如,把程序 ex05_16 改为至少答对三题才能结束.

```
PROGRAM ex05_17
! trivia quiz.
!
DO
  READ question$, answer$
  PRINT question$;
  LINE INPUT reply$
  IF reply$=answer$ then
     LET right=right+1
     PRINT"correct"
```

```
  ELSE
     PRINT"No, the correct answre is";answer$;"."
  END IF
  LET num_quest=num_quest+1 !count questions
  IF end data and right<3 then
     RESTORE
     LET right, num_quest=0
     PRINT"Plase again"
  END IF
LOOP until end data
PRINT"You got";100*right/num_quest;"%right."
DATA What is the capital of Austria, Vienna
DATA What year did Franklin pierce take office, 1853
DATA"How many years, on average, does a baboon live", 20
DATA How about a gray squirrel, 5
END
```

5.5 循环结构程序设计举例

这一节想通过一些实例说明循环结构语句的使用，以及有关的一些程序设计方法和技巧，以引导读者能编出高质量的程序.

例 1 求 $\sum_{n=1}^{8} n!$ 的值.

解 设和为 sum，从求 sum 出发，第一步可以描述为：

① sum⟸0

② for n=1 to 8

　　计算 n!

　　sum⇐sum+n!

　next n

③ 打印输出 sum

第二步 逐步具体化，进一步确定 n!

① sum=0

② for n=1 to 8

　　n1⇐n1*n (n1 初值为 0!=1)

　　sum⇐sum+n1

　next n

③ 打印输出 sum

至此问题已解决. 这种"自顶向下,逐步求精"的方法,即为结构化程序设计方法. 这种类似于BASIC语句, 而又夹带了文字叙述的描叙, 就叫伪语言代码. 它比程序语言更接近于自然语言. 这样完全细化了的描述,再转化为True BASIC也不难了. 依据上述分析可编程序如下:

```
PROGRAM ex05_18
!
LET sum=0
LET n1=1
FOR n=1 to 8
      LET n1=n1*n
      PRINT n,"n!=";n1
      LET sum=sum+n1
NEXT n
PRINT"sum=";sum
END
```

例 2 编写一个程序,求下式之和:

$$sum=\sum_{i=1}^{20}\sum_{j=1}^{3}(j+1)^2(i+2)$$

解 这个问题比较简单. 求两次累加和,需要在一重循环内嵌套一重循环,叫二重循环. 可编程序如下:

```
1   PROGRAM ex05_19a
2   !
3   LET sum=0
4   FOR i=1 to 20
5         LET wk=i+2
6         FOR j=1 to 3
7               LET sum=sum+wk*(j+1)*(j+1)
8         NEXT j
9   NEXT i
10  PRINT"sum=";sum
11  END
```

在程序中,求sum的基本操作是第 7 行. 而循环附加操作有:外循环赋初值 1 次,循环终结判断 20 次;内循环赋初值 20×1 次,循环终结判断 20×3 次,附加操作共 101 次.

尽管多重循环的基本操作次数大体上是一个定值,但为完成循环控制所需的附加操作次数,却因循环的具体结构不同而呈现出较大的差别. 例如, 根据加法交换定律, 则内外两重循环交换,结果不变. 可编程序如下:

```
PROGRAM ex05_19b
!
LET sum=0
```

```
4  FOR j=1 to 3
5      LET wk=(j+1)∧2
6      FOR i=1 to 20
7          LET sum=sum+wk*(i+2)
8      NEXT i
9  NEXT j
10 PRINT"sum=";sum
11 END
```

在这个程序中,完成循环控制所需的附加操作有:外循环 j 赋初值 1 次,判断循环终结 3 次;内循环赋初值 3 次,判断循环终结 3×20 次,附加操作 67 次,减少 34 次. 通过这个例子说明,在循环程序设计中,应遵守如下原则:

① 与循环控制变量无关的运算,应放到循环之前, 如程序 ex05_19a 和程序 ex05_19b 的第 5 行,不应该放在第 7 行内.

② 如果内外层循环允许交换,则应把控制变量变化范围大的放在内层.

例 3 试编写一个模拟学习评分程序. 学习内容是: 任意两位以内整数加法. 评分规则是每次 10 题,当第 1 次回答正确时得 10 分,否则不得分. 为了学习每题可答数次,答对为止.

解 从评分规则出发,第一步可作如下描述:

```
① result⇐0                (成绩赋初值)
② for i=1 to 10
      随机生成两位以内整数 a,b
      DO
         显示题目
         回答 c
         计算回答次数
         if a+b=c then
            print "评语"
            if 第一次回答 then result⇐result+10
         else
            print "评语"
         end if
      Loop 直到 a+b=c
   next i
③ 打印结果
```

其中,随机生成某一范围的整数, 这是一个小的技巧. 随机函数 rnd 的值是 $0\leqslant rnd<1$, 所以,要用放大和加减的方法变化随机数的范围. 如生成 1 到 6 之间的数, 可用 int((6*rnd)+1) 来表示, 这里可用 int(100*rnd) 生成小于 100 的整数. 根据上述分析,可编程序如下:

```
PROGRAM ex05_20
!
LET result=0
RANDOMIZE
FOR i=1 to 10
      LET p=0
      LET a=int(100*rnd)
      LET b=int(100*rnd)
      DO
           IF p>=3 then EXIT DO
           PRINT i;" ";a;"+";b;"=";
           INPUT c
           LET p=p+1
           IF int(a+b)=int(c)then
              PRINT tab(15);p;tab(20);
              PRINT"That's right, Very good!"
              IF int(p)=1 then LET result=result+10
           ELSE
              PRINT tab(15);p;
              PRINT"No, try again."
           END IF
      LOOP until int(a+b)=c
NEXT i
PRINT" result=";result
END
```

例 4 找出 100 到 200 之间的全部素数.

解 初等数学定义:只能被 1 和本身除尽的数叫素数. 这个问题可描述为:

```
for i=100 to 200
  if i 是素数 then
    打印 i
  end if
next i
```

判断 i 是否是一个素数,不是一个逻辑表达式能描述的,而是需要一个过程. 为表示判定结果应该引入一个逻辑变量,由于 True BASIC 没有逻辑类型变量,所以,要设定一个指示器变量 P,规定当 P=1时表示 i 是素数,当 P=0 时表示 i 不是素数. 设置指示器变量是常用的程序设计方法之一. 这样,从素数定义出发,验证 i 的过程可描述如下.

```
p<=1
for k=2 to i-1
  if mod(i,k)=0 then p<=0
```

```
next k
```

这样可编程序如下:

```
PROGRAM ex05_21a
!
FOR i=100 to 200
  LET p=1
  FOR k=2 to i-1
       IF mod(i,k)=0 then LET p=0
  NEXT k
  IF p=1 then PRINT i
 NEXT i
 END
```

关于循环参数的选择问题,这是提高效率的重要技巧之一. 上述程序有两重循环,外层循环中,i 为偶数显然不是素数,这样可改为

```
FOR i=101 to 199 step 2
```

内层循环也不必用偶数去除,可改为

```
FOR k=3 to i-1 step 2
```

其实,k 的终值只要小于 $\sqrt{i}$ 的最大整数即可. 如 i=101 取 k 的终值为 10 即可. 因此 k 循环可改为

FOR k=3 to $\sqrt{i}$ step 2

根据上述分析可编程序如下:

```
PROGRAM ex05_21b
!
FOR i=101 to 199 step 2
       LET p=1
       FOR k=3 to sqr(i) step 2
           IF mod(i,k)=0 then LET p=0
       NEXT k
       IF p=1 then PRINT i,
NEXT i
 END
```

运行结果:

101	103	107	109	113
127	131	137	139	149
151	157	163	167	173
179	181	191	193	197
199				

习 题

5.1 计算 π，已知公式如下：

$$\frac{\pi^2}{6}=1+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{n^2}+\cdots\cdots$$

要求精度为$\frac{1}{n^2}<10^{-6}$.

5.2 按下式计算 sin x：

$$\sin x=\frac{x}{1!}-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\cdots-\frac{x^{2n-1}}{(2n-1)!}+\frac{x^{2n+1}}{(2n+1)!}-\cdots\cdots$$

计算到最后一项的绝对值小于 10^{-5} 为止.

5.3 用欧几里得算法(辗转相除法)，求两个正整数的最大公约数.

5.4 用牛顿迭代法求$\sqrt[3]{a}$. 设 a=3，要求精确度为小数点后第四位.

[提示] 牛顿迭代公式为(设 f(x)=0)

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$$

5.5 试问 100! 的末尾有多少个零?

[提示] 统计因子 5 出现的个数.

5.6 有一头母牛，每年年初生一头小母牛，每头小母牛从第四年起每年年初也生一头小母牛，问在第 20 年时，共有多少头母牛。

[提示]数列为 2, 3, 4, 6, 9, ….

通式为 $u_n=u_{n-1}+u_{n-3}\quad(n\geqslant4)$.

5.7 在马克思数学手稿中，有这样一个题目：有钱 50 先令，请客 30 人. 每个男人要用 3 先令，每个女人要用 2 先令，每个小孩要用 1 先令. 问主人有多少请客方案? 男人、女人和小孩各有多少?

5.8 有职工 n 人，假设每人工资没有小于元的零钱，打印一张分发工资的清单(要求发给每人的张数最少. 如，某人工资 59 元，则他应得 10 元 5 张，5 元 1 张，2 元 2 张).

5.9 编写一个程序，键入任意一个整数，检查它是奇数(odd)还是偶数(even)，若整数是大于 1 的奇数，再确定它是否为质数.

5.10 编写一个程序，键入一个八进制的数字串，把它转换为十进制数. 例如，输入 74203，输出 Suppleid integer(base8):74203

Decimal eguivalent:30875

5.11 编写一个程序，打印出以下图形：

```
* * * * * * * * *
  * * * * * * *
    * * * * *
      * * *
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
```

5.12 编写一个程序，打印出以下图形：

```
          *
         * *
        * * *
       * * * *
      * * * * *
     * * * * * *
```

第 6 章　数组及其应用

6.1 数组概述

6.1.1 数组变量的引入

迄今为止，我们在程序中用到的变量都是简单变量，就是说每一个变量有一个名字，有一个值．用简单变量处理如下问题是很不方便的．

例如，处理一张如表 6-1 所示的学生成绩单．

表 6-1

学　号	数学	语文	政治	物理	化学	外语	平均分
1	85.2	80.5	91.0	94.2	85.4	96.0	
2	60.5	90.6	85.5	85.5	74.6	85.1	
3	71.2	85.4	76.5	76.5	66.4	70.4	
⋮							
35	99.0	65.4	90.0	90.0	87.5	75.4	

学生成绩单是因人和时间而异的，所以成绩单的每一项都是一个变量，它的值都是数值．

又如，一张干部基本情况表，包括姓名、性别、年龄、职称、籍贯通信地址等，这些项也是因人因时而异的，每一项都是一个变量，它的值可以用字符串表示．

在数学上有向量和矩阵，分别记作：

$$C=(c_1,c_2,c_3,\cdots,c_n)$$

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

数学上的向量和矩阵分别也是一组变量，它的每个元素的值是数值．

为解决上述这类问题，True BASIC 引入了数组．数组是具有相同类型的有序变量的集合．例如，数组 C 的元素分别记作 C(1),C(2),…,C(n)，即是带下标的变量，所以数组元素也叫“下标变量”．除特殊情况外，下标变量和简单变量一样可作各种处理，如赋值、运算、输入或输出等．本章不再一一重述．

从结构化的角度来说，数组是一个结构化的变量，为了便于处理数组变量（简称数组），True BASIC 引入了一组 MAT 语句，MAT 是 MATRIX（矩阵）的缩写．这组语句可以对数组作整体处理，如读、写、赋值、加、减、求转置矩阵、求逆矩阵等．本章将以向量和矩阵为主讲述上述这些语句及其应用．

6.1.2 数组说明

编程序让计算机处理数组，首先要说明数组的属性，包括名称、类型、维数、容量等，然后才能进行处理，即“先说明，后处理”.

（1）数组说明符的一般形式:

数组名（维1，维2，…，维n）

其中，数组名：用符号名表示，带$为字符串类型，否则为数值类型.

维：可以是，下界 to 上界或上界（这时，下界为1）.

界：可以是任意数（正数、负数或0），下界≤上界.

n：表示维数. IBM PC 机 n≤255. 常用的是一维、二维和三维，如C(3),B(1949 to 1988, 2),A(3,2,2).

True BASIC 规定：数组说明符可以出现在MAT READ, MAT INPUT 和 MAT LINE INPUT 语句中，但一定要先说明，即首先要出现在数组说明语句中.

（2）数组说明语句的一般形式:

DIM 数组说明符 1，数组说明符 2，…

其中，DIM 为关键字，是 DIMENSION 的缩写. 例如

DIM c(3),b(3,2),a(3,2,2),weather$(12,31)

分别说明数组属性，如表 6-2 所示.

表6-2

数组说明符	数组名	类 型	维 数	容 量
c(3)	c	数值	1	3
b(3,2)	b	数值	2	6
a(3,2,2)	a	数值	3	12
weather$(12,31)	weather$	字符串	2	372

6.1.3 下标变量（数组元素）

（1）下标变量的一般形式:

数组名（下标表达式 1，下标表达式 2，…）

其中，下标表达式是数值表达式，计算后自动四舍五入取整. 例如 c(2),c(3),b(2,1),weather$(5,1).

注意，下标变量和数组说明符有时形状一样，但其本质不同. 下标变量只是代表数组的一个元素. 在程序中用各自出现的位置去区别，数组说明符只能出现在 MAT READ,MAT INPUT,MAT LINE INPUT 和 DIM 语句之中，而下标变量则出现在其它语句之中. 如赋值语句、表达式等，所以不会弄混.

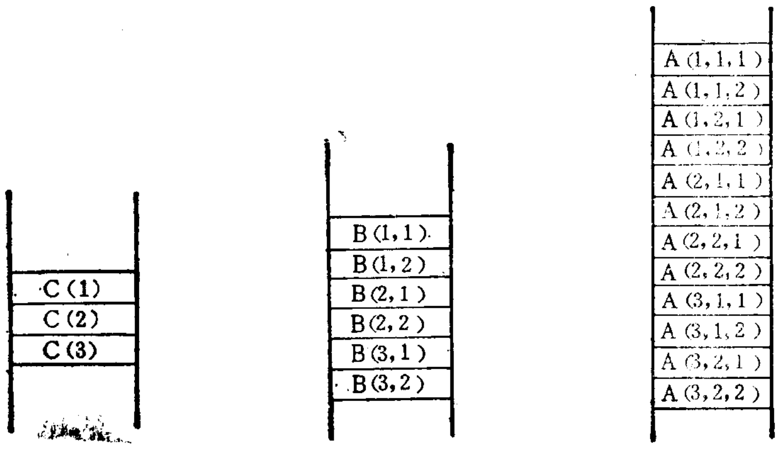
（2）数组的存储.

简单变量是名字、值、存储单元一一对应的，而数组是一个名字有若干个数组元素，它们的值在存储器中是按下标值的大小顺序连续排列的. 数组元素的排列位置、计算公式，如表 6-3 所示.

表 6-3

维　数	数组说明符	数组元素	数组元素的排列位置	最大下标值
1	c(n)	c(k)	k	n
2	b(m,n)	b(j,k)	(j−1)·n+k	m·n
3	a(L,m,n)	c(i,j,k)	(i−1)·m·n+(j−1)·n+k	L·m·n

例如，有数组 C(3)，B(3,2)，A(3,2,2) 在存储器中的排列顺序如图 6-1 所示.



(a)　　(b)　　(c)

图 6-1　数组元素存储示意图

总之，数组元素在存储器中的顺序很象汽车计程器的运转，如数组 A(3,2,2)，由 A(1,1,1)，A(1,1,2) 到 A(3,2,2). 因此，True BASIC 数组元素在存储器中的顺序称为"按计程器顺序".

如果作个形象的比喻，下标可分别取名如下：

一维数组　aname(列)

二维数组　aname(行,列)

三维数组　aname(页,行,列)

四维数组　aname(册,页,行,列)

五维数组　aname(卷,册,页,行,列)

这时，每个数据相当于书中的一个字符. 数据的顺序正是和人们阅读字符的顺序一样.

注意，下标变量要和数组说明符保持一致，即名字、维数一样. 下标表达式的值不能超界，否则会出错误.

6.2 数组赋值与运算

6.2.1 数组赋值

在 True BASIC 中引入了数组赋值语句，一般形式如下:

MAT a=aexpr (1)

MAT a=(expr) (2)

其中， a 为数组变量.

aexpr 为数组表达式(见 6.2.2)或数组.

expr 为表达式，如果是常数或简单变量可以去掉括号.

执行赋值语句(1)是把一个数组表达式的值(一个数组)赋给数组变量 a; 执行赋值语句 (2) 是把一个数或一个字符串赋给数组变量的各元素，各元素值是相同的. 赋值号两边类型要一致。在赋值语句(1)的赋值号两边，数组的维数也要一致. 例如

```
PROGRAM ex06_1
!
DIM a(5,5),b(5,5),c$(6)
FOR i=1 to 5
      LET b(i,i)=1
NEXT i
MAT a=b
MAT c$=("no"&"ne")
END
```

6.2.2 数组运算

True BASIC 引入数组运算的目的是为了简化数学上的矩阵运算，所以数组运算也叫矩阵运算，一切规则和数学上的矩阵运算规则一致. 比如，多维矩阵可以加减，二维矩阵可以求逆、求转置、相乘，但不能相除等，此外 True BASIC 还规定: 矩阵表达式只能含一个运算符. 如

```
MAT a=b+c
MAT a=b-c
MAT a=b*c
MAT a=n*c
```

其中 a,b,c 是矩阵，n 是常数.

对于加法和减法，数组 b 和数组 c 必须维数相同，每一维的上、下界可以不同，但它包含元素的个数要相等. 如数组 b 是 3×4，数组 c 也应是 3×4.

例 1

```
PROGRAM ex06_2
!
DIM a(5),b(5),sum(5),dif(5)
```

```
MAT a=1
MAT b=5
MAT sum=a+b
MAT dif=b-a
END
```

例 2 试编写一个程序，验证矩阵乘法运算.

解 设 a(l,m)，b(m,n)，c(l,n)，数学规定矩阵乘法规则是：

$$c_{ik}=\sum_{j=1}^{m}a_{ij}b_{jk},\qquad \begin{array}{l} i=1,2,\cdots,l \\ j=1,2,\cdots,m \\ k=1,2,\cdots,n \end{array}$$

按照这个规则可编程序如下：

```
1  PROGRAM ex06_3
2  !
3  DIM a(2,3),b(3,4),c(2,4),d(2,4)
4  MAT a=1
5  MAT b=1
6  MAT d=a*b
7
8  FOR i=1 to 2
9      FOR k=1 to 4
10         LET c(i,k)=0
11         FOR j=1 to 3
12             LET c(i,k)=c(i,k)+a(i,j)*b(j,k)
13         NEXT j
14     NEXT k
15 NEXT i
16
17 LET tp=1
18 FOR i=1 to 2
19     FOR j=1 to 4
20         IF abs(d(i,j)-c(i,j))>1e-6 then
21            LET tp=0
22         END IF
23     NEXT j
24 NEXT i
25 IF tp=1 then PRINT"good!" else PRINT"bad!"
26 END
```

程序的第 6 行和第 8～15 行效果一样，从而可以看出，引入矩阵运算后，程序更简洁. 注意，数组元素象简单变量一样可以逐个比较，而数组与数组是不能直接比较的.

6.2.3 系统数组(内部数组)

在数组处理中,经常有赋初值的问题,比如数组初值是单位矩阵(即主对角线元素值为 1,其它元素为 0),要分别对元素赋值,书写起来也比较麻烦,为此,True BASIC 引入了系统数组. 系统数组的值在系统中准备好了,可供随时引用. True BASIC 的系统数组如表 6-3 所示.

表 6-3

矩 阵 名	含 义
Zer或Zer(l,m,…)	数组元素值皆为 0
Con或Con(l,m,…)	数组元素值皆为 1
Idn或Idn(l,l)	主对角线元素值为 1,其它为 0
Nul\$或Nul\$(l,m,…)	数组元素皆为空串

其中 l,m,…为数值常数.

用这些数组可对其它数组赋值,或乘以常数再赋值,但不能直接参加矩阵运算. 使用有确定界限的系统数组给数组变量赋值,可以改变数组变量的大小(详见 6.4).

6.2.4 有关数组的函数

因为数组的大小是可调的,为了知道数组的大小、各维的上界与下界, True BASIC 引入了以数组名为参数的一组函数——数组函数.

(1) Size(a)　　在数组 a 中元素的个数.

(2) Size(a,n)　　在数组 a 中第 n 维元素的个数.

(3) Lbound(a)　　一维数组 a 的下界.

(4) Ubound(a)　　一维数组 a 的上界.

(5) Lbound(a,n)　　数组 a 第 n 维的下界.

(6) Ubound(a,n)　　数组 a 第 n 维的上界.

除上述之外, True BASIC 还引入了如下矩阵函数:

(7) Trn(x)　x 为 n×n 矩阵,函数值为矩阵 x 的逆矩阵.

(8) Inv(x)　x 为 m×n 矩阵,函数值为矩阵 x 的转置矩阵.

(9) Det(x)　x 为 n×n 矩阵,函数值为一个数,是方阵行列式的值. 如 Det(x)是 0, 或接近于 0,则方阵 x 是奇异阵或接近奇异阵. 如果 x 为奇异阵,求 x 的逆阵,将出现运行错误.

(10) Dot(x,y)　x,y 为具有相同元素的向量,函数值为 x,y 的内积(数性积),即

$$\text{Dot}(x,y)=\sum_{i=1}^{n}x_iy_i$$

6.3 数组的输入/输出语句

为简化数组的输入/输出，True BASIC 引入了 MAT 输入/输出语句.

6.3.1 MAT PRINT 语 句

一般形式:

MAT PRINT array1,array2,…

MAT PRINT array1;array2;…

其中 array 为数组名.

执行 MAT PRINT 语句可显示出各数组的所有元素. 每个数组按“计程器顺序”显示数组元素. 每当一维变化结束时，光标自动指向新的一行开始处，因此，二维数组正是按人们希望的一行一行地显示，在每个数组之后显示一个空行. 对于三维或更多维数组，每二维之间也显示一个空行，好象一页一页的分开显示.

类似于 PRINT 语句，以一个逗号结尾的数组名，它的每个元素按区域格式显示；以一个分号结尾的数组，它的元素按紧凑格式显示. 最后一个数组名如果不用逗号或分号结尾，则表示用逗号结尾. 例如

```
PROGRAM ex06-4
!
DIM a(10), one(2, 2), two(3, 3), three(2, 3, 4)
FOR i=1 to 10
    LET a(i)=i
NEXT i
MAT one=1
MAT two=2
MAT threc=3
MAT PRINT a;one, two, three;
END
```

运行结果:

```
1   2   3   4   5   6   7   8   9   10
1                     1
1                     1
2                     2                 2
2                     2                 2
2                     2                 2
3   3   3   3
3   3   3   3
3   3   3   3
```

3　　3　　3　　3
3　　3　　3　　3
3　　3　　3　　3

6.3.2 MAT PRINT USING 语句

MAT PRINT USING 语句是 MAT PRINT 和 PRINT USING 语句的结合. 它按一个共同的格式显式数组元素. 一般形式为:

MAT PRINT USLNG format$:array1,array2,…

其中 format$ 和在 PRINT USING 语句中的 format$ 一样, 自由选择各种格式. 如果元素个数超出了格式串范围,格式串将从头开始起被重复使用.

和使用 MAT PRINT 语句一样, 数组是一行一行的输出. 如果用一条 MAT PRINT USING 语句显示几个数组时,在数组之间输出一个空行. 例如

```
PROGRAM ex06_5
!
DIM a(3,4),b(6)
FOR i=1 to 3
     FOR j=1 to 4
          LET a(i,j)=10*i+j
     NEXT j
NEXT i
MAT b=zer
MAT PRINT using"##.#   $##      ":a,b
END
```

运行结果:

```
11.0 $12 13.0 $14
21.0 $22 23.0 $24
31.0 $32 33.0 $34

 .0 $0   .0 $0   .0 $0
```

6.3.3 MAT INPUT 语句

一般形式:

MAT INPUT array1,array2,…

MAT INPUT PROMPT expr$:array1,array2,…

其中 expr$ 是字符串表达式,作为待输入项的提示.

执行这个语句和 INPUT 语句相似,可以从键盘上按"计程器顺序"对各个数组赋值,各输入数据项之间用逗号隔开. 各数组的数据之间用一个回车换行(即 enter 键)隔开,其它规定和 INPUT 语句一样. 例如

```
PROGRAM ex06_6
!
```

```
DIM a(6), b(2, 3, 4)
MAT INPUT a, b
PRINT"    Thank you."
MAT PRINT a, b
END
```

运行结果:

```
? 1, 2, 3, 4, 5, 6
? 111, 112, 113, 114, 121, 122, 123, 124, 131, 132, 133, 134
TOO few input items. Please add more.
? 211, 212, 213, 214, 221, 222, 223, 224, 231, 232, 233, 234
    Thank you.
1          2          3          4          5
6
111        112        113        114
121        122        123        124
131        132        133        134
211        212        213        214
221        222        223        224
231        232        233        234
```

6.3.4 MAT LINE INPUT语句

一般形式:

MAT LINE INPUT array1$, array2$, …

MAT LENE INPUT PROMPT expr$:array1$, …

其中 expr$ 是一个字符串表达式,作为待输入项的提示.

MAT LINE INPUT 是 MAT INPUT 和 LINE INPUT 语句的组合,执行它能从键盘上输入足够多的行送给数组的每个元素,数组元素是按"计程器顺序"存入的.其它和 LINE INPUT 语句一样.例如

```
PROGRAM ex06_7
! Ask for a list of favorite phrases.
DIM phrase$(3)
MAT LINE INPUT phrase$
PRINT
PRINT"Okay, as I understand it, you like:"
PRINT
FOR i=1 to 3
    PRINT phrase$(i)
NEXT i
END
```

运行结果:

- You make a better door than you do a window.
- Dull as ditchwater.
- That's not a bug, it's a feature.

Okay, as I understand it, you like:

You make a better door than you do a window.
Dull, as ditchwater.
That's not a bug, it's a feature.

6.3.5 MAT READ语句

一般形式:

```
MAT READ array1, array2, …
  ⋮
DATA item11, item12, …
DATA item21, item22, …
```

其中item是array的数据,按"计程器顺序"排列.

执行MAT READ语句可以从DATA语句取数据项送给数组元素,它的作用和READ语句类似.

例 求解n阶方程组

$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=b_1 \\ a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n=b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n=b_n \end{cases}$$

解 设

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix},\quad X=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},\quad B=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

则

$$A\cdot X=B,\quad X=A^{-1}\cdot B$$

设n=7,根据上述分析可编程序如下:

```
PROGRAM ex06_8
!
DIM a(7,7), a_1(7,7), b(7), x(7)
! mat b=1
! mat a=zer
MAT READ a, b
IF det(a)>=1e-6 then
```

```
8      MAT a_1=Inv(a)
9      MAT x=a_1*b
10     MAT PRINT using"##.###":x
11  ELSE
12     PRINT"a is illigell data."
13  END IF
14  DATA  12.418,  -1.061,   2.669,   4.361,  -0.119,  -1.209,  -0.500
15  DATA  -1.501,  19.832,   0.694,  -4.816,   2.274,   2.001,  -1.909
16  DATA   2.308,   1.728, -15.165,  -2.023,   1.104,   2.107,  -1.000
17  DATA   3.359,  -0.913,  -6.441,  27.864,   3.737,  -4.375,  -2.375
18  DATA  -1.562,   1.168,  -2.004,   1.818,   9.490,   0.401,  -1.073
19  DATA   1.174,   7.318,  -2.278,  -0.143,  -9.835, -31.670,   4.114
20  DATA   0.109,  -1.313,  -0.900,  -1.972,  -3.514,  -1.107,  12.094
21  DATA   8.262, -33.818, -52.673, -97.284,  20.351, 149.918,  81.653
22  END
```

运行结果:

```
1.000   -2.000   3.000   -4.000   5.000   -6.000   7.000
```

这个程序用了求逆矩阵函数，使程序变得非常简洁. 还用了求矩阵行列式的值来判断是否是奇异矩阵，保证了程序安全运行.

6.4 可调数组

如前所述，True BASIC 规定:每个数组在使用前，必须用 DIM 语句给予说明，以确定数组名、维数和各维的上下界. 有的计算机语言(如 1975 年公布的标准 PASCAL)规定数组一经说明不准改变，即固定数组. 有的语言(如 FORTRAN)规定在辅程序中可以改变数组的上下界，即局部可调数组. 使用固定数组常常出现数组存储空间的过剩或不足. 为此 True BASIC 规定:在主程序中经 DIM 语句说明的数组，它的上下界是可以改变的，同时在辅程序中数组的上下界也是可以改变的. 因而，True BASIC 的数组是真正的可调数组. 调整方式分两类: 一类是显式调整，一类是隐式调整.

6.4.1 显式调整

可用如下语句来实现:

MAT READ array1,array2,…

MAT INPUT array1,array2,…

MAT LINE INPUT array1,array2,…

其中 array 为各维的下界或上界可以是

〈数值表达式〉 (下界保持不变)

〈数值表达式 1〉 to 〈数值表达式 2〉

此外

MAT INPUT array(?)

是允许的.

在输入数据时,如果一行内数据项未完,可以用逗号结束表示,换行后继续输入,否则回车换行表示输入结束. 例如

```
PROGRAM ex06_9
! redimension
DIM a(1), b(2, 3)
MAT INPUT a(?)
MAT READ b(2, 2)
DATA 1, 2, 3, 4
MAT PRINT a;b
PRINT"a has"; ubound(a);"element."
END
```

运行结果:

```
? 11, 22, 33, 44, 55,
? 66, 77
11 22 33 44 55 66 77

1            2
3            4

a has 7 element.
```

6.4.2 隐式调整

可用矩阵赋值语句实现:

MAT Aname = Aexpr

其中 Aname 为数组名.

Aexpr 为数组表达式(包括简单运算,系统数组等).

执行赋值语句,可以改变 Aname 各维的上界. 注意,Aexpr 和 Aname 的维数必须一致,否则会出现错误信息.

例 1 任意给定 n 个数,求其中最小值.

解 算法设计:

第一步,把 n 个数输入到一个数组中.

第二步,设置一个工作单元,供存放最小值用,并赋初值.

第三步,用工作单元中的值和数组中的元素逐个地进行比较,若数组元素小,则留下来,即存入工作单元;否则不存入,继续比较. 根据算法可编程序如下:

```
PROGRAM ex06_10
!
```

```
DIM x(1)
MAT INPUT x(1)
LET small=x(1)
FOR i=2 to ubound(x)
    IF x(i)<small then LET small=x(i)
NEXT i
PRINT
PRINT small
END
```

例 2 学生成绩简单统计表. 求每个学生的总分与平均分.

解 设表的第 1 项为学号, 第 2～第 l+1 项为各科成绩, 第 l+2 项为总和, 第 l+3 为总平均成绩. 可编程序如下:

```
PROGRAM ex06_11
DIM tabl(2, 2)
READ n, l
MAT tabl=zer(n, l+3)
MAT READ tabl
FOR i=1 to n
    FOR j=2 to l+1
        LET tabl(i, l+2)=tabl(i, l+2)+tabl(i, j)
    NEXT j
    LET tabl(i, l+3)=tabl(i, l+2)/l
NEXT i
MAT PRINT using"####.#":tabl
DATA 6, 4
DATA 1, 98.0, 65.8, 78.4, 65.3, 0, 0
DATA 2, 89.4, 67.0, 98.1, 56.6, 0, 0
DATA 3, 56.7, 89.7, 78.4, 63.0, 0, 0
DATA 4, 89.5, 100.0, 78.5, 98.8, 0, 0
DATA 5, 68.5, 98.0, 67.7, 98.0, 0, 0
DATA 6, 65.0, 89.0, 74.7, 68.6, 0, 0
END
```

运算结果:

1.0	98.0	65.8	78.4	65.3	307.5	76.9
2.0	89.4	67.0	98.1	56.6	311.1	77.8
3.0	56.7	89.7	78.4	63.0	287.8	71.9
4.0	89.5	100.0	78.5	98.8	366.8	91.7
5.0	68.5	98.0	67.7	98.0	332.2	83.0
6.0	65.0	89.0	74.7	68.6	297.3	74.3

6.5 数 组 应 用

例1 试编一个程序，把一个数组元素按升序排列，并把它打印出来.

解 在计算机科学与应用中，排序又叫分类. 分类的目的是便于随后对已分类的集合元素进行检索. 这是一个极重要的问题，不少人作过深入的研究，有许多算法. 这里只介绍如下两种方法.

方法一，直接选择分类法. 它的基本思想是，选择小的数往前排，如图 6-2 所示.

```
55  33  88  99  11
11  33  88  99  55
11  33  88  99  55
11  33  55  99  88
11  33  55  88  99
```

图 6-2 直接选择分类法一例

具体地说，第一次在 5 个数中，选最小的排在第一，第二次在余下的 4 个数中选择最小的排在第二，如此等等. "选择"的基本操作是比较、交换.

假设某数组有 10 个元素，根据上述分析，可画出如图 6-3 所示的 N-S 图.

根据图 6-3，可编程序如下：

```
PROGRAM ex06_12a
!
DIM x(10)
MAT INPUT x
FOR i=i to 9
      LET min=i
      FOR j=i+1 to 10
            IF x(j)<x(min) then LET min=j
      NEXT j
      LET wk=x(i)
      LET x(i)=x(min)
      LET x(min)=wk
NEXT i
MAT PRINT x;
END
```

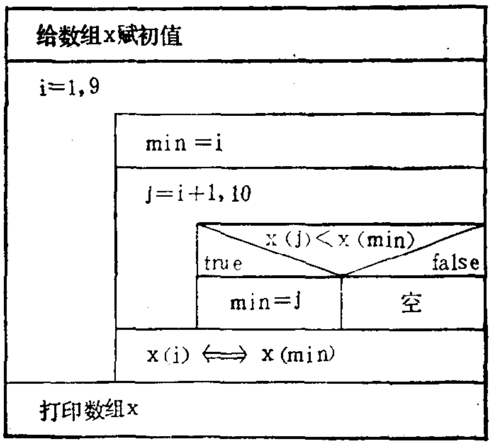


图 6-3 直接选择分类法 N-S 图

方法二，直接交换分类法. 它的基本方法是，比较并交换相邻的元素对，直到所有元素都被放到正确的地方为止，如图 6-4 所示.

33	88	55	66	11
33	55	66	11	88
33	55	11	66	88
33	11	55	66	88
11	33	55	66	88

图 6-4 直接交换分类法一例

具体地说，通过相邻元素的比较，第一次把 5 个数中最大的排在倒数第一，第二次把余下 4 个数中最大的排在倒数第二，如此等等. 这个方法也叫大数下沉法.

假设某数组有 10 个元素，根据上述分析可画出如图 6-5 所示的 N-S 图.

根据图 6-5，可编程序如下：

```
PROGRAM ex06_12b
!
DIM x(10)
MAT INPUT x
FOR i=2 to 10
      FOR j=1 to 10+1-i
            IF x(j)>x(j+1) then
               LET wk=x(j)
               LET x(j)=x(j+1)
               LET x(j+1)=wk
            END IF
      NEXT j
NEXT i
MAT PRINT x;
END
```

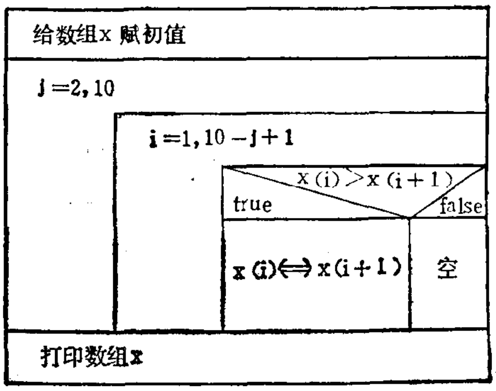


图 6-5 直接交换分类法 N-S 图

例 2 输入若干英文单词，按字典顺序排序.

解 用直接交换法，这里把二重循环写成如下形式：

```
FOR i=n to 2 step -1
    FOR j=1 to i-1
        ......
        ......
    NEXT j
NEXT i
```

其含义是一样的. 可编程序如下：

```
PROGRAM ex06_13
! Program to sort text in a list.
DIM name$(1)
PRINT"    Enter items to be sorted"
MAT INPUT name$(?)        ! Redimension to input
```

```
LET n=Ubound(name$)          ! How many we have
FOR i=n to 2 step -1
    FOR j=1 to i-1
        IF name$(j)>name$(j+1) then
           LET temp$=name$(j)  ! Temporary variable
           LET name$(j)=name$(j+1)
           LET name$(j+1)=temp$
        END IF
    NEXT j
NEXT i
MAT PRINT name$                ! Now print sorted list
END
```

运行结果:

```
   Enter items to be sorted
? by, one, if, two,
? input, read, run, end
by             end        if         input        one
read           run        two
```

在这个程序中,用了可调数组.

例3 学生成绩统计表. 求每个学生各科总分和平均分,并按学生成绩从高到低打印出来(若分数相同前后任意).

解 这个问题的解法是把 ex06-12a 和 ex06-11 结合起来. 可编程序如下:

```
PROGRAM ex06_14
DIM tabl(2, 2)
READ n, l
MAT tabl=zer(n, l+3)
MAT READ tabl
FOR i=1 to n
    FOR j=2 to l+1
        LET tabl(i, l+2)=tabl(i, l+2)+tabl(i, j)
    NEXT j
    LET tabl(i, l+3)=tabl(i, l+2)/l
NEXT i
FOR i=1 to n-1
    LET max=i
    FOR j=i+1 to n
        IF tabl(j, l+3)>tabl(max, l+3) then
           LET max=j
        END IF
    NEXT j
```

```
        FOR k=1 to l+3
            LET wk=tabl(i, k)
            LET tabl(i, k)=tabl(max, k)
            LET tabl(max, k)=wk
        NEXT k
    NEXT i
    MAT PRINT using'####.#':tabl
    DATA 6, 4
    DATA 1, 98.0, 65.8, 78.4, 65.3, 0, 0
    DATA 2, 89.4, 67.0, 98.1, 56.6, 0, 0
    DATA 3, 56.7, 89.7, 78.4, 63.0, 0, 0
    DATA 4, 89.5, 100.0, 78.5, 98.8, 0, 0
    DATA 5, 68.5, 98.0, 67.7, 98.0, 0, 0
    DATA 6, 65.0, 89.0, 74.7, 68.6, 0, 0
    END
```

运行结果:

4.0	89.5	100.0	78.5	98.8	366.8	91.7
5.0	68.5	98.0	67.7	98.0	332.2	83.0
2.0	89.4	67.0	98.1	56.6	311.1	77.8
1.0	98.0	65.8	78.4	65.3	307.5	76.9
6.0	65.0	89.0	74.7	68.6	297.3	74.3
3.0	56.7	89.7	78.4	63.0	287.8	71.9

例 4 打印如下九九表:

** A Multiplecation Table **

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
(1)	1								
(2)	2	4							
(3)	3	6	9						
(4)	4	8	12	16					
(5)	5	10	15	20	25				
(6)	6	12	18	24	30	36			
(7)	7	14	21	28	35	42	49		
(8)	8	16	24	32	40	48	56	64	
(9)	9	18	27	36	45	54	63	72	81

解 生成九九表的数组是比较容易的,关键是要恰当地使用PRINT语句和 PRINT USING 语句把数据打印到合适的位置上. 程序如下:

```
PROGRAM ex06_15
DIM itab(9, 9)
PRINT
PRINT
PRINT tab(9);'** A Multiplecation Table**'
```

```
PRINT
PRINT "    ";
FOR i=1 to 9
    PRINT using"(#)":i;
NEXT i
PRINT
FOR i=1 to 9
    FOR j=1 to i
        LET itab(i,j)=i*j
    NEXT j
NEXT i
FOR i=1 to 9
    PRINT using"(#)":i;
    FOR j=1 to i
        PRINT using"#####":itab(i,j);
    NEXT j
    PRINT
NEXT i
END
```

例5 计算多项式

$$f(x)=a_1x^n+a_2x^{n-1}+a_3x^{n-2}+\cdots+a_nx+a_{n+1}$$

的值. 设 x, n, 以及所有系数 $a_1, a_2, \cdots, a_{n+1}$ 均为已知.

解 方法一，直接法. 它是直接计算每一项的值，然后相加，可用如下程序段实现:

```
PROGRAM ex06_16a
! to calculate the value of polynomial
 DIM a(1)
INPUT prompt"x,n=":x,n
PRINT "input";n+1;"term"
MAT INPUT a(n+1)
LET f=0
FOR i=1 to n
    LET term=a(i)
    FOR j=1 to n-i+1
        LET term=term*x
    NEXT j
    LET f=f+term
NEXT i
LET f=f+a(n+1)
PRINT x,f
END
```

用这种算法，计算第一项要作 n 次乘法，计算第二项要作 $(n-1)$ 次乘法，…，计算第 n 项要作一次乘法，合计共需作 $n+(n-1)+\cdots+2+1=0.5n(n-1)$ 次乘法，或约为 n^2 次乘法.

解 方法二，秦九韶法. 秦九韶是我国宋代数学家，最先提出下述算法. 五六百年之后，霍纳(Horner)也提出了同样算法，国外称为 Horner 法. 秦九韶把 f(x) 多项式化作

$$f=(\cdots(a_1x+a_2)x+\cdots+a_n)x+a_{n+1}$$

得到递推公式：

$$f_1=a_1$$

$$f_j=f_{j-1}x+a_j \quad (j=2,3,\cdots,n+1)$$

则 $f=f_{n+1}$. 若将 $f_1,f_2,f_3,\cdots,f_{n+1}$ 看作是变量 f 的值，则迭代公式为

$$f=0$$

$$f=fx+a_j \quad (j=1,2,3,\cdots n+1)$$

根据上述思想，可编程序如下：

```
PROGRAM ex06_16b
! to calculate the value of polynomial
DIM a(1)
INPUT prompt"x, n=":x, n
PRINT"input";n+1;"term"
MAT INPUT a(n+1)
LET f=0
FOR i=1 to n+1
      LET f=f*x+a(i)
NEXT i
PRINT x, f
END
```

用秦九韶方法计算多项式，只需作 n+1 次乘法，比 n^2 次少了 n 倍，效率大大提高.

例 6 有个农民在菜园周围种了 l 棵萝卜，并按顺时针方向给每棵萝卜一个代码. 若农民从第一号开始拔，然后，沿顺时针方向每隔两棵拔一棵，直到拔完为止. 试编一个程序，按拔萝卜顺序打印出萝卜代码的数列.

解 设 l=12，选用两个数组 m(12) 和 n(12). 用 m(12) 表示菜园，它的初值为代码：1,2,3,…,12. 若被拔掉，则冲 0，表示萝卜已不存在. 用 n(12) 顺序记下所拔萝卜的代码. 拔萝卜是沿着菜园重复进行的，转一圈即循环一次，循环次数不知道，拔完是结束条件. 这个问题，包含了两重循环，内层是从 1 到 12 逐个地检查是否已被拔掉，再数一、二、三，…. 外层是达到拔完为止，即拔掉棵数为 12 时，结束循环. 因此，内层是记数型循环，外层为条件型循环. 根据上述分析，可画 N-S 图如图 6-6 所示.

根据 N-S 图可编程序如下：

```
PROGRAM ex06_17
DIM m(12),n(12)
```

```
FOR i=1 to 12
    LET m(i)=i
NEXT i
LET k=2
LET j=0
DO
    FOR i=1 to 12
        IF m(i) <> 0 then
           LET k=k+1
           IF k >=3 then
              LET j=j+1
              LET n(j)=m(i)
              LET m(i),k=0
           END IF
        END IF
    NEXT i
LOOP until j>=12
MAT PRINT n;
END
```

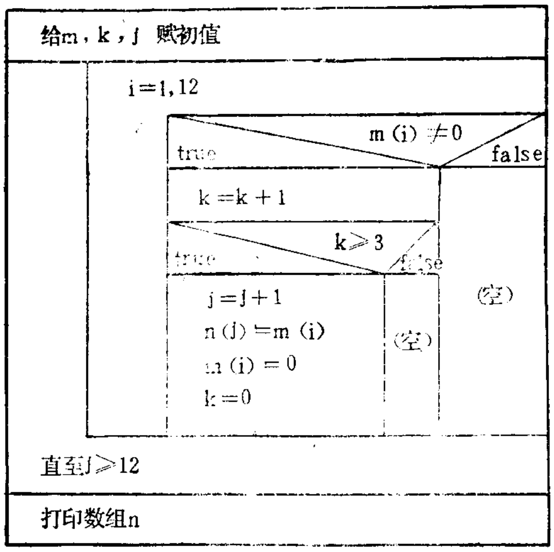


图 6-6

运行结果:

1 4 7 10 2 6 11 5 12 9 3 8

习 题

6.1 给定任意一个数组(方阵), 求它的逆矩阵、转置矩阵和矩阵元素个数.

6.2 利用随机函数产生 10 个 10 到 50 之间的随机整数, 按紧凑格式打印出这组数值, 并选出这 10 个数中最大的一个数.

6.3 利用随机函数产生 20 个 200 到 300 之间的随机整数, 按紧凑格式打印出这组数值, 再把这 20 个数按大小顺序排列之后打印出来.

6.4 利用随机函数产生 50 个 200 到 300 之间的随机整数. 首先按紧凑格式打印这组数值, 然后选出其中的素数, 并把这些素数按大小顺序排列输出.

6.5 已知任一方阵, 求方阵的下三角形中各元素之和.

6.6 已知任一方阵, 求出方阵的最大元素和对应的两个下标.

6.7 每人打 5 发子弹, 9 名学生实弹射击成绩如下表:

学生号码	每发子弹射中环数				
307	6	8	8	9	8
814	8	7	0	10	8
253	6	10	8	6	7
900	4	5	3	8	7
435	6	7	5	5	7
526	0	10	10	10	10

224	1	4	6	9	7
109	6	0	4	8	10
480	5	6	7	9	8

要求:

(1) 打印出人均总环数.

(2) 打印成绩最好和最差的学生号码及对应的总环数. 但若有一发子弹射中 0 环者,不能评为最好.

6.8 已知任一个二维数组,求出各行元素之和,按和的大小重新排序,并打印出各行原来的行号.

6.9 编写一个程序,能求任一行或任一列元素之和.

6.10 已知任一个二维数组,找出其中最大元素,然后在最大元素对应行和列元素不变的条件下,把最大元素移到下标为(1,1)的位置.

6.11 为方便摄氏温度(C)与华氏温度(F)的转换:

(1) 请按下列格式打印一张摄氏与华氏温度的对照表. 列为摄氏 36～40 度,行距为摄氏 0.1 度. 计算公式为: $F=\frac{9}{5}C+32$

	36	37	38	39	40
0.0	…	…	…	…	…
0.1	…	…	…	…	…
0.2	…	…	…	…	…
⋮					
0.9	…	…	…	…	…

表格中的数值为对应的华氏温度,取小数点一位,之后的小数四舍五入.

(2) 用 INPUT 语句输入摄氏温度 38.4,执行程序,从对照表中找到对应的华氏温度,并打印结果:

38.4C……××.×F

6.12 计算并打印简单工资统计表. 工资表头用汉语拼音的缩写:职工号(ZGH)、基本工资(JBGZ)、附加工资(FJGZ)、补助(BZ)、洗理费(XLF)、房租费(FZF)、水电费(SDF)、托儿费(TEF)、病事假扣除(BSJKC),计算每个职工的纯收入以及各项的总计.

第7章 图 形

True BASIC 不仅可以作科学计算和事务处理，还可以画图，而且有很强的功能. 本章将介绍画点、画线、画椭圆，画多边形，在图上写文字和着色等基本功能，以及显示动画的功能. 除此之外，True BASIC 还可以对图画作各种变换，这留在 8.7 节再介绍.

7.1 图形坐标

显示屏是由称为"象素"的磷光点覆盖的，它能发光并显示多种色彩. 和文字输出一样，在画图程序的控制下，在屏幕上可以显示出各种图形. 通常在显示屏上分编辑窗口和背景窗口，在执行画图程序时，编辑窗口和背景窗口会自动消失，整个显示屏变为图形窗口. 为确定图形各点在图形窗口上的位置，和机械制图一样，在图形窗口设置一个直角坐标系(x, y)，这里称为图形坐标.

7.1.1 图形坐标设置语句

SET WINDOW $x_{min}, x_{max}, y_{min}, y_{max}$

其中，x_{min} 为窗口上 x 轴的最小值.

x_{max} 为窗口上 x 轴的最大值.

y_{min} 为窗口上 y 轴的最小值.

y_{max} 为窗口上 y 轴的最大值.

例如:

```
SET WINDOW -14,14,-10,10
SET WINDOW 0,1,0,1
```

如图 7-1 所示.

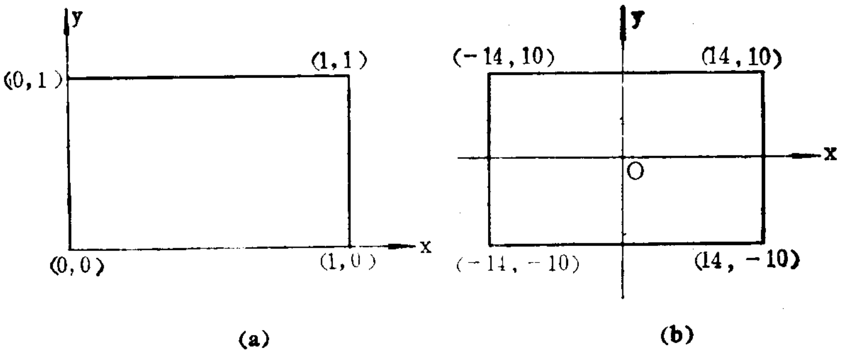


图 7-1 图形坐标

当给定图象任一点的坐标时，True BASIC 会自动地把它转化为象素地址. 这样程序员无需知道任何象素知识，可以象机械制图一样在显示屏上画图. 如果在画图程序中，图形坐标设置语句被缺省，这时，系统约定图形坐标如图 7-1(a)所示.

图形坐标可以重新设置. 如果想在同一个窗口画出几个图形，而且图形的直角坐标系不一样，这时不必用坐标变换统一坐标系，True BASIC 允许在程序的任何位置重新设置坐标系，而且，窗口上原有图形保持形状不变.

7.1.2 询问图形坐标语句

为了知道当前图形坐标的位置，True BASIC 引入了询问坐标语句:

ASK WINDOW $x_{min}, x_{max}, y_{min}, y_{max}$

其中 $x_{min}, x_{max}, y_{min}, y_{max}$ 为数值变量.

执行它，系统将把当前坐标 x，y 的最小值和最大值分别赋给 $x_{min}, x_{max}, y_{min}, y_{max}$.

说明: 纵横比. 图形坐标的设置直接影响画图的描述以及图形的好坏. 因为 y 与 x 方向单位长度比(称为纵横比)不恰当，可能会出现"画圆不圆，画方不方"的现象. 要适当调整图形坐标设置语句的参数，即可得到满意的结果. 通常选用 $(y_{max}-y_{min}):(x_{max}-x_{min})=5:7$，即纵横比为 5:7.

7.2 基本画图语句

7.2.1 画点

(1) 画点语句一般形式:

PLOT POINTS: x1, y1 (叫定点无向输出)

或 **PLOT x1, y1** (True BASIC)

MAT PLOT POINTS: array

其中 array 是 n×2 的二维数组，即 n 个点的坐标表，一点占一行，如数组 points(100,2).

(2) 应用举例.

例如，用点画一条正弦曲线，可编程序如下:

程序 1:

```
PROGRAM ex07_1a
! plot sine values
!
SET window 0, 2*pi, -1, 1
FOR x=0 to 2*pi step pi/100
      PLOT POINTS: x, sin (x)
NEXT x
SET window 0, 4*pi, -1, 1
FOR x=0 to 4*pi step pi/100
```

```
        PLOT POINTS:x, sin(x)
NEXT x
END
```

程序 2:

```
PROGRAM ex07_1b
! plot sine values
!
DIM points(201, 2)
SET window 0, 2*pi, -1, 1
FOR x=0 to 2*pi step pi/100
        LET k=k+1
        LET points(k, 1)=x
        LET points(k, 2)=sin(x)
NEXT x
MAT PLOT POINTS:POINTS
END
```

ex07-1a 运行结果如图 7-2 所示.

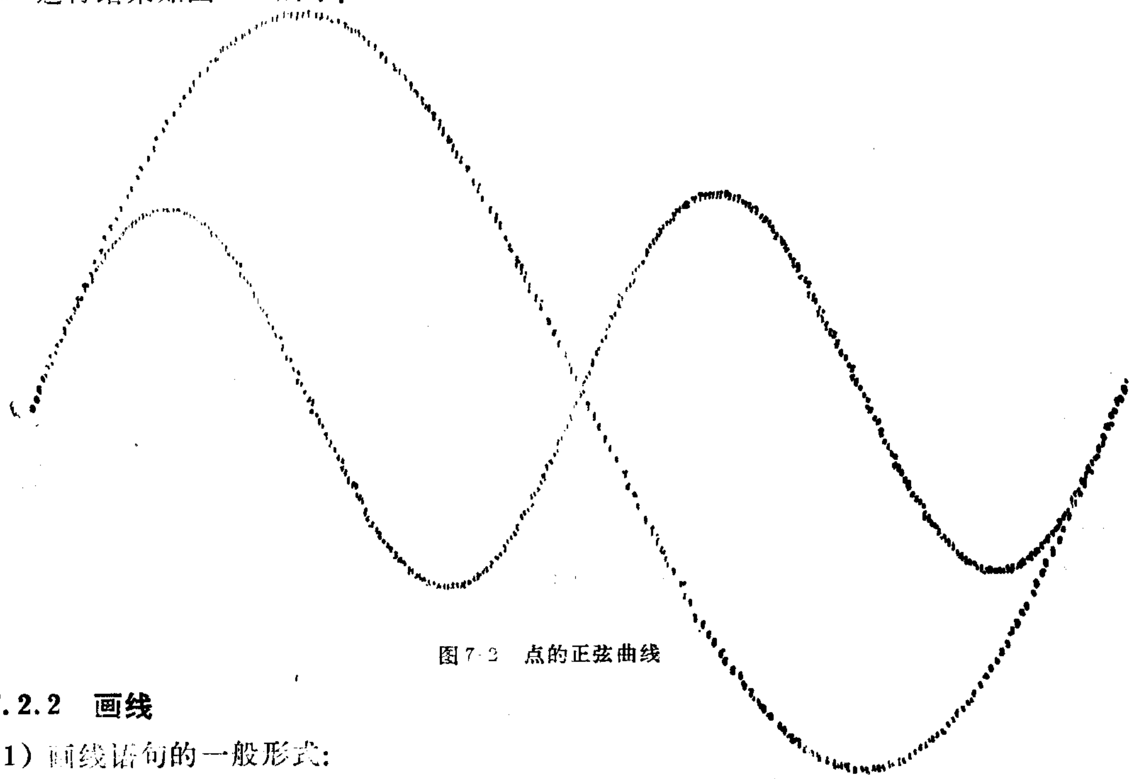


图 7-2 点的正弦曲线

7.2.2 画线

(1) 画线语句的一般形式:

PLOT LINES: x1, y1; (叫定点有向输出)

或 **PLOT x1, y1;** (True BASIC)

MAT PLOT LINES: array

其中 array 为 n 个点的坐标表,如 Line(100, 2).

(2) 应用举例.

例如，画一条余弦曲线，可编程序如下：

程序 1:

```
PROGRAM ex 07_2a
!  plots a cosine curve
!
DIM line (201,2)
SET window 0, 2*pi, -2, 2
PLOT LINES: 0, 0; 2*pi, 0
PLOT LINES: 0, 2; 0, -2
FOR x=0 to 2*pi step pi/100
     PLOT lines: x, cos(x);
NEXT x
END
```

程序 2:

```
PROGRAM ex07_2b
! plots a cosine curve
!
DIM line (201,2)
SET window 0, 2*pi, -2, 2
PLOT LINES: 0, 0; 2*pi, 0
```

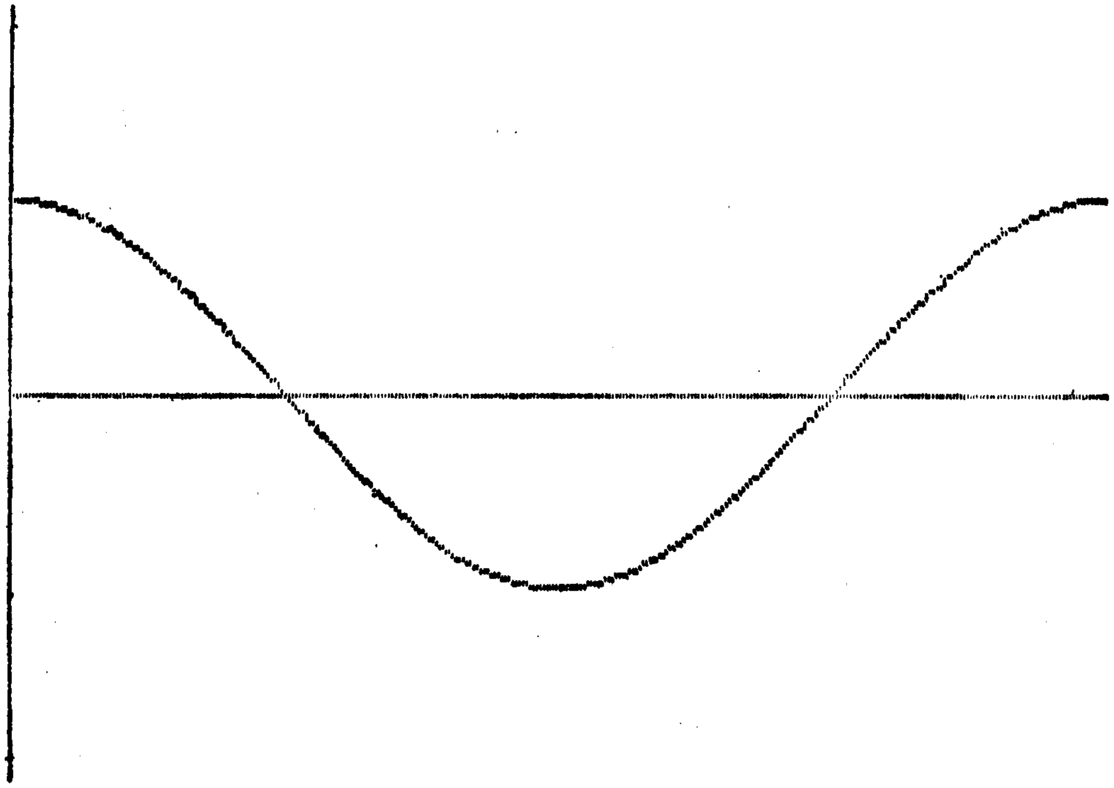


图 7-3　余弦曲线

```
PLOT LINES: 0, 2; 0, -2
FOR x=0 to 2*pi step pi/100
    LET p=p+1
    LET Line (p, 1)=x
    LET Line (p, 2)=cos(x)
NEXT x
MAT PLOT LINES:Line
END
```

运行结果如图 7-3 所示.

7.2.3 画面

(1) 画面语句的一般形式:

PLOT AREA: $x_0, y_0; x1, y1; \cdots; x_n, y_n$

MAT PLOT AREA: array

其中 array 为 n 个点的坐标表,如 area(200, 2).

执行它, 光束从第 1 点开始画一条线到第二个点,然后,从第二个点画一条线到第三个点, 如此等等, 直到从最后一点返回第 1 个点形成一个封闭的曲线,然后,用当前颜色(如绿色) 填满整个区域.

(2) 举例.

例 1 画一个五角星.

解 选取图形坐标如图 7-4 所示.

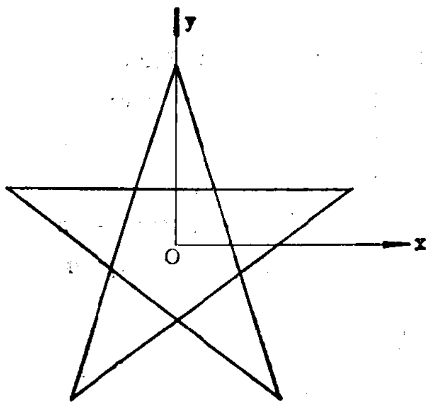


图 7-4 五角星与坐标

利用极坐标与直角坐标的转换关系,可编程序如下:

```
PROGRAM ex 07_3
DIM x(0 to 4),y(0 to 4)
SET window -14, 14, -10, 10
LET bgn=pi/2
LET pio=4*pi/5
FOR k=o to 4
    LET x(k)=8*cos(bgn+k*pio)
    LET y(k)=8*sin(bgn+k*pio)
NEXT k
PLOT AREA:x(0), y(0);x(1), y(1);
        x(2), y(2); x(3), y(3); x(4); y(4)
END
```

运行结果如图 7-5 所示.

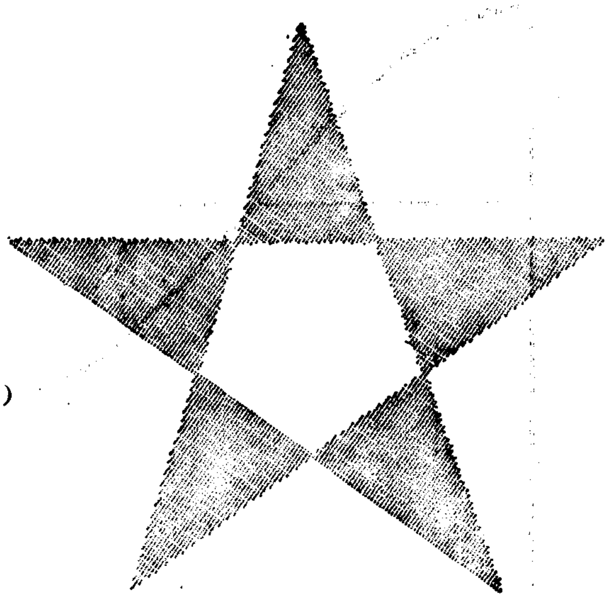


图 7-5 五角星

这里有一个疑问,为什么图 7-5 中五角星内部的五角形内部是背景颜色,原因是这部分区域在曲线围成的区域之外.

例 2 画里沙玉图形.

解 里沙玉公式

$$\begin{cases} x = p\sin(a\varphi) \\ y = q\cos(b\varphi) \end{cases}$$

依据这个公式可编程序如下:

```
PROGRAM ex07_4
! draw a Lissajous figure
!
DIM points(201, 2)
SET window -1, 1, -1, 1
INPUT prompt"p, q, a, b? " : p, q, a, b
CLEAR
 FOR i=0 to 2*pi step pi/100
      LET c=c+1
      LET points (c, 1)=p*sin(a*i)
      LET points (c, 2)=q*cos(b*i)
NEXT i
MAT PLOT AREA : points
END
```

当 $p=1, q=1, a=3, b=5$ 时,运行结果如图 7-6 所示.

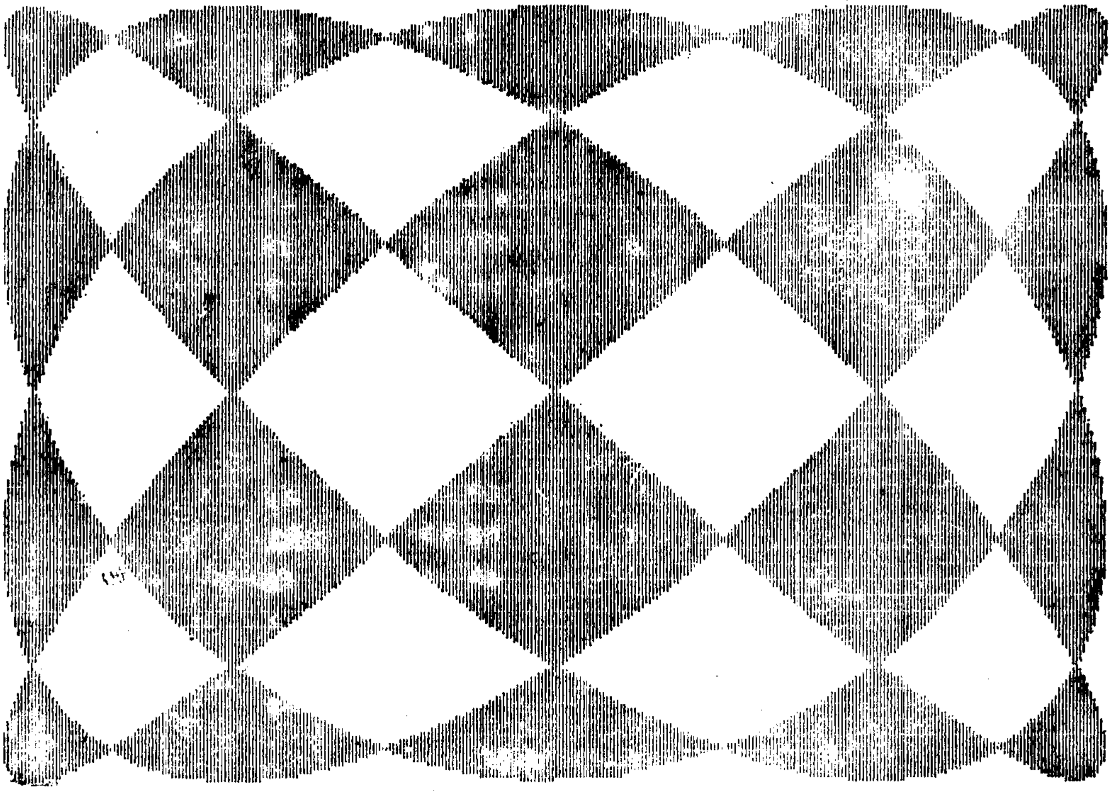


图 7-6　里沙玉图形

7.2.4 在图形坐标上字符的输出

(1) PLOT TEXT 语句的一般形式:

PLOT TEXT, AT x, y: expr$

其中 x, y 为一个点的图形坐标. expr$ 为字符串表达式.

执行它,把 expr$ 显示在(x, y) 指定的图形坐标处. 它和 PRINT 语句不同的是:输出位置是用图形坐标指定的.

例 绘制直方图.

解 下面以绘制学生成绩分布图为例,可编程序如下:

```
PROGRAM ex07_5
!
SET window 0, 6, 0, 60
FOR x=i to 5
     READ value
     BOX AREA x-0.4, x+0.4, 0, value
     PLOT TEXT, at x-0.4, Value+3:Using$("###.#", value)
NEXT x
DATA 5.1, 21.5, 32.0, 40.5, 10.9
END
```

运行结果如图 7-7 所示.

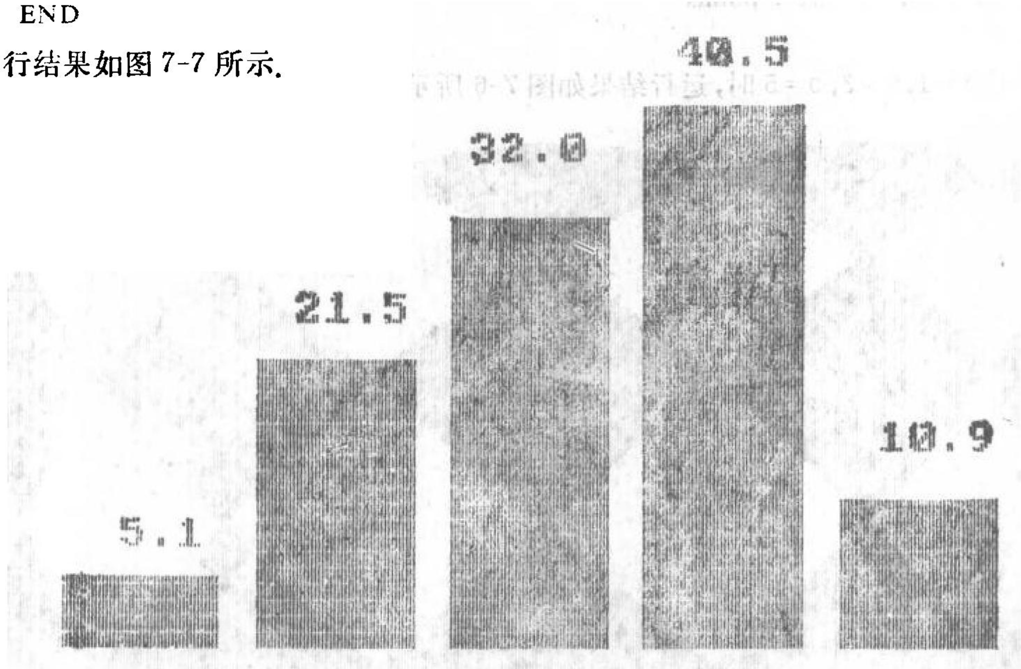


图 7-7 学生成绩分布图

注: ① USING$ 为标准函数,详见表 3-2, 用于指定格式输出. ② BOX AREA语句画矩形面(见7.4.2).

7.3　图形颜色的设置与询问

至此，我们画的图形以及显示的字符(统称为前景)都是绿色，而背景(后景)颜色是黑色. 其实，True BASIC 允许用户设置前景颜色和背景颜色，若没有设置即缺省，则系统既定了前景颜色为绿色，背景颜色为黑色.

7.3.1　前景颜色的设置与询问

前景颜色有两种色调模式：一种是绿、红、黄；另一种是深兰、深红、白. 同一种色调模式的三种颜色可以同时显示. 前景颜色如表 7-1 所示.

表 7-1

code	expr$	含　义	code	expr$	含义
0	"background"	背景颜色			
1	"green"	绿色	4	"cyan"	深兰
2	"red"	红色	5	"magenta"	深红
3	"yellow"	黄色	6	"white"	白色

注：如前景颜色取 0 或"background"，这时前景颜色与背景颜色一致.

前景颜色可用如下语句设置与询问：

SET COLOR expr$　(用字符串设置当前颜色)

SET COLOR code　(用代码设置当前颜色)

ASK COLOR var$　(询问当前颜色)

ASK COLOR var　(询问当前颜色代码)

其中 code 为数字表达式，值为颜色代码. var为变量名.

除此之外，还能用如下语句询问最多能显示几种前景颜色：

ASK MAX COLOR var

其中 var 为变量名.

如果新设置的前景颜色不属当前色调模式，则改变当前色调模式，同时已有图形的颜色将按如下对应关系改变：

green←→cyan

red←→magenta

yellow←→white

7.3.2 后景颜色

后景颜色如表 7-2 所示.

表 7-2

code	expr$	含义	code	expr$	含 义
0	“black”	黑色	8		灰色
1	“blue”	兰色	9		明亮兰
2	“green”	绿色	10		明亮绿
3	“cyan”	深兰	11		明亮深兰
4	“red”	红	12		明亮红
5	“magenta”	深红	13		明亮深红
6	“brown”	棕色	14	“yellow”	黄
7	“white”	白色	15		明亮白

注：Code 还可以取值 16～31 和 0～15 对应，数字意义相同，只是加强了前景颜色的亮度，即增加了对比度. 后果颜色的选择与前景颜色无关，即不会引起前景颜色的改变.

后景颜色可用如下语句设置和询问：

SET BACK expr$　　(用字符串设置后景颜色)

SET BACK code　　(用代码设置后景颜色)

ASK BACK var$　　(询问背景颜色)

ASK BACK var　　(询问背景颜色代码)

其中 BACK 也可写作 BACKGROUND COLOR(美国国家标准)意思更清楚，但比较麻烦.

7.4 BOX 语 句

True BASIC 为实现快速画图还引入一组 BOX 语句. BOX 是用 x_{min}，x_{max}，y_{min}，y_{max} 表示一个盒子，这组语句都与“盒子”有关. 可以画矩形框、矩形面、椭圆(包括圆). 除此之外，还可以用显示与删除语句产生动画效果.

7.4.1 画矩形框

画矩形框语句的一般形式：

BOX LINES x_{min}，x_{max}，y_{min}，y_{max}

其中 x_{min}，x_{max}，y_{min}，y_{max} 为图形坐标.

例如，由大到小画 10 个不同颜色的方框，程序如下：

```
PROGRAM ex07_6
! draw ten boxes
!
SET window -14, 14, -10, 10
FOR i=10 to 1 step -1
      SET color mod (i, 2)+1
      BOX LINES -i, i, -1, i
```

```
NEXT i
END
```

运行结果如图 7-8 所示.

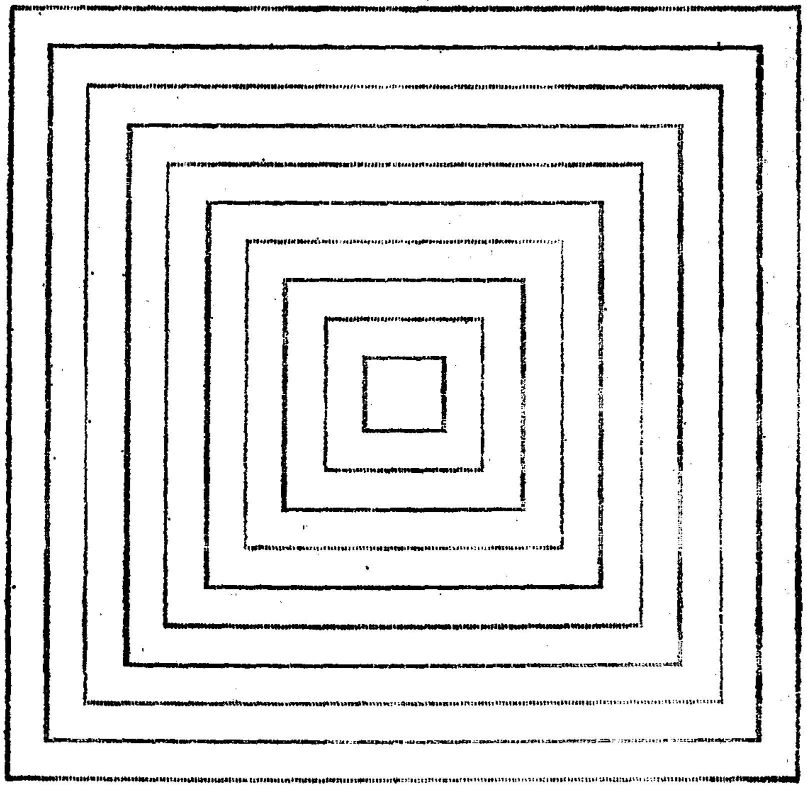


图 7-8

7.4.2 画矩形面

画矩形面语句的一般形式:

BOX AREA $x_{min}, x_{max}, y_{min}, y_{max}$

其中 $x_{min}, x_{max}, y_{min}, y_{max}$ 为图形坐标.

例如,画由大到小 10 个方块,程序如下:

```
PROGRAM ex07_7
! draw ten boxes
!
SET window -14, 14, -10, 10
ASK max color num
FOR i=10 to 1 step -1
    SET color mod (i, num) +1
    BOX AREA -i, i, -i, i
NEXT i
END
```

运行结果如图 7-9 所示.

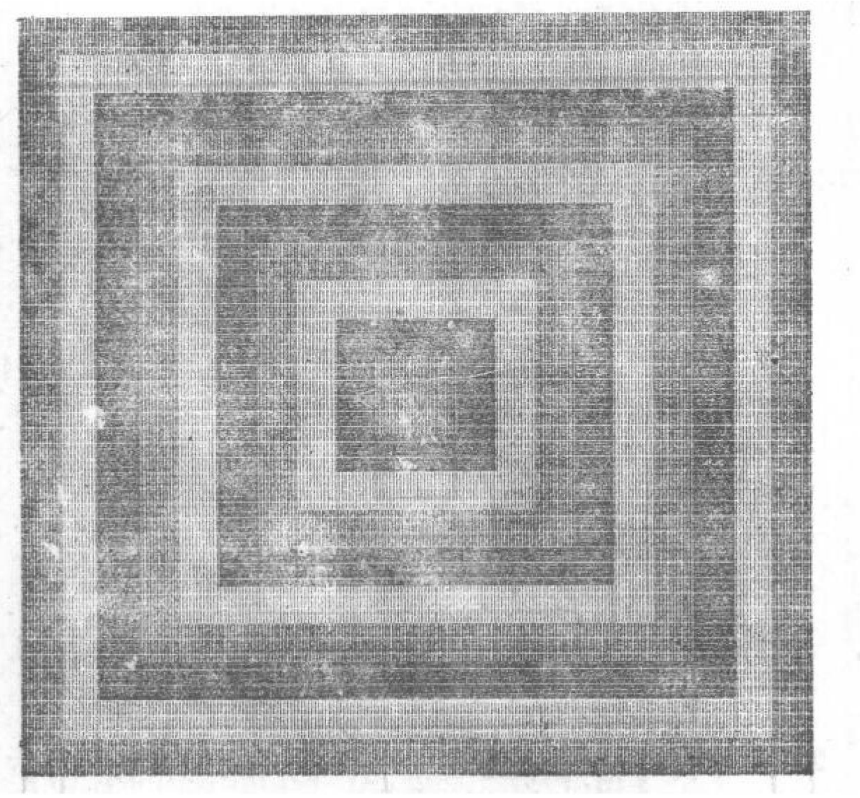


图 7-9

7.4.3 画椭圆(或圆)

画椭圆语句的一般形式:

BOX ELLIPSE $x_{min}, x_{max}, y_{min}, y_{ma}$

或 **BOX CIRCLE** $x_{min}, x_{max}, y_{min}, y_{max}$

这两个语句功能相同,都能画椭圆和圆.

例如,画由大到小 10 个圆,程序如下:

```
PROGRAM ex07_8
! Bull's eye pattern.
!
SET window -14, 14, -10, 10
FOR i=10 to 1 step -1
      SET color mod (i, 3)+1
      BOX CIRCLE -i, i, -i, i
NEXT i
END
```

运行结果如图 7-10 所示.

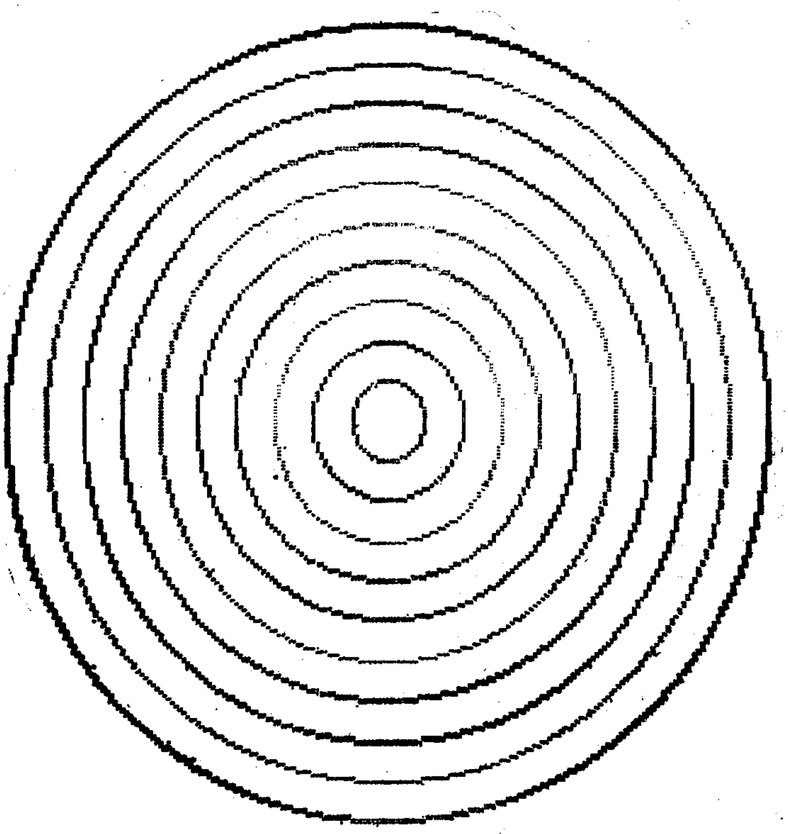


图 7-10

7.4. FLOOD 语句

一般形式:

FLOOD x,y

其中 x, 为图形坐标下的任一点的坐标.

执行它,点(x,y)所在的区域充满当前前景颜色. 如点 (x,y) 是圆内一点,则圆内充满当前前景颜色. 例如,程序

```
PROGRAM ex07_9
  Bull's eye pattern.
!
SET window -14,14,-10,10
FOR i=10 to 1 step -1
    SET color mod(1,3)+1
    BOX CIRCLE -i,i,-i,i
    FLOOD 0,0
NEXT i
PAUSE 3
END
```

这个程序是以坐标原点为中心画圆,然后,充满当前前景颜色, 半径由大到小连续画 10 次,

运行结果如图 7-11 所示.

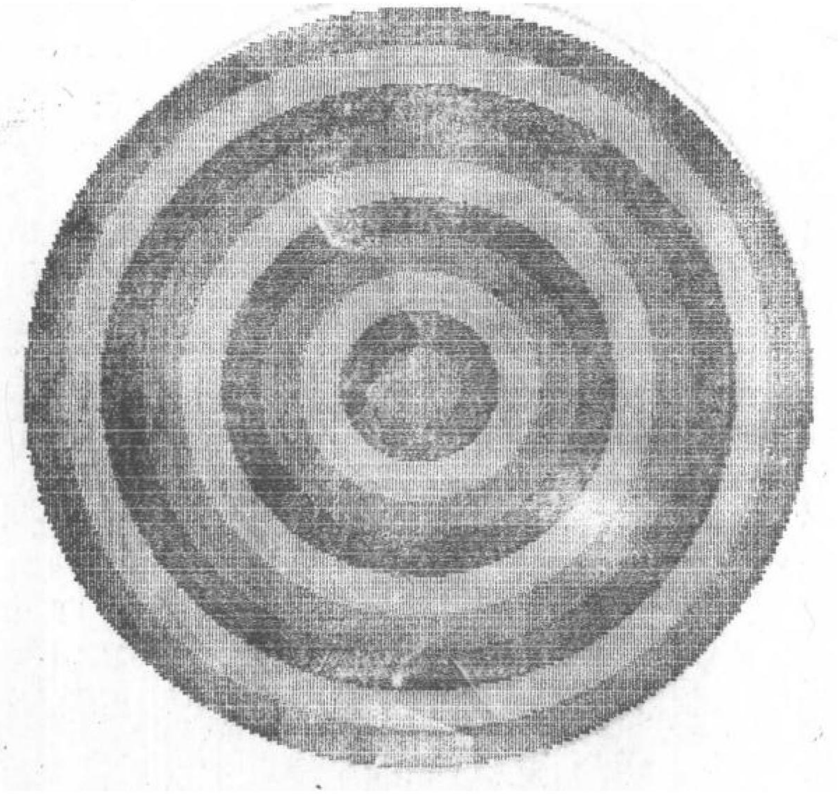


图 7-11

7.4.5 动画

True BASIC 为实现动画功能,引入如下保存、删除和显示图形的三个语句:

BOX KEEP $x_{min}, x_{max}, y_{min}, y_{max}$ **in var$**

执行它,把盒子内图形(包括前景和后景)存入一个字符串变量之中.

BOX CLEAR $x_{min}, x_{max}, y_{min}, y_{max}$

执行它,删除盒子内的图形,即用后景颜色填满盒子.

BOX SHOW var$ AT x_{min}, y_{min}

其中 x_{min}, y_{min} 为任一点的图形坐标.

执行它,以(x_{min}, y_{min})为盒子的左下角坐标,显示存储在 var$ 中的图形.

例 小球沿正弦曲线滚动表演,程序如下:

```
PROGRAM ex07_10
! a bouncing ball.
!
SET window -14, 14, -10, 10
BOX CIRCLE -1, 1, -1, 1
FLOOD 0, 0
BOX KEEP -1, 1, -1, 1 in ball $
BOX CLEAR -1, 1, -1, 1
```

```
9   !
10  SET window 0,2*pi,-2,2
11      PLOT 0,0;2*pi,0
12      PLOT 0,-2;0,2
13      FOR x=0 to 2*pi step pi/200
14          PLOT LINES:x, sin(x);
15      NEXT x
16      FOR x=0 to 2*pi step pi/200
17          BOX SHOW ball$ at x, sin(x)
18          BOX CLEAR x,x+0.5, sin(x), sin(x)+0.5
19          PLOT LINES:x, sin(x);
20      NEXT x
21  END
```

这个程序的第 4～8 行，是画一个小球以及保存它，抹掉它的过程. 第 10～20 行是画一个坐标系，然后就是小球沿正弦曲线滚动过程. 运行某一时刻显示如图 7-12 所示.

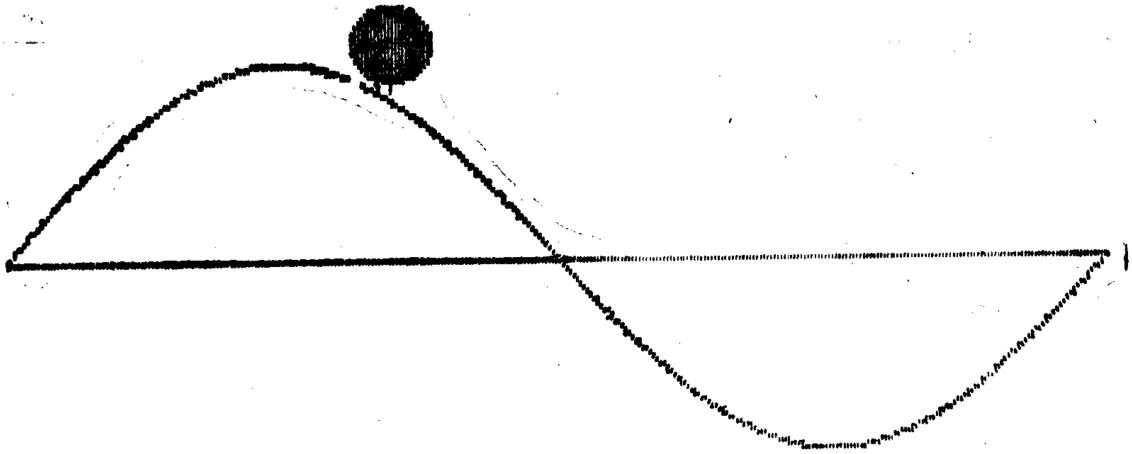


图 7-12

此外，**BOX SHOW** 还有两种扩展形式:

BOX SHOW var$ AT x_{min}, y_{min} USING expr$

BOX SHOW var$ AT x_{min}, y_{min} USING expr

其中，USING expr$ 和 USING expr 部分，给定了 var$，存储的图象同屏幕上已有图象按位组合的方式.

expr$ 可以是“and”，“or”或“xor”，含义见表 7-3.

expr 是数字表达式，取 mod(expr,16)为组合方式代码，代码含义如表 7-3 所示.

注意: BOX SHOW 的这些扩展形式，不能由一个计算机转到另一个计算机上，因为它们依赖于图象在存储器中存放的形式，在不同型号的计算机上，它们有完全不同的结果，请参看所使用的计算机用户指南.

表 7-3

表达式值	存储的值	屏幕的值	新的值
0	0 0 1 1	0 1 0 1	0 0 0 0
1(“and”)	0 0 1 1	0 1 0 1	0 0 0 1
2	0 0 1 1	0 1 0 1	0 0 1 0
3	0 0 1 1	0 1 0 1	0 0 1 1
4	0 0 1 1	0 1 0 1	0 1 0 0
5	0 0 1 1	0 1 0 1	0 1 0 1
6 (xor)	0 0 1 1	0 1 0 1	0 1 1 0
7 (or)	0 0 1 1	0 1 0 1	0 1 1 1
8	0 0 1 1	0 1 0 1	1 0 0 0
9	0 0 1 1	0 1 0 1	1 0 0 1
10	0 0 1 1	0 1 0 1	1 0 1 0
11	0 0 1 1	0 1 0 1	1 0 1 1
12	0 0 1 1	0 1 0 1	1 1 0 0

续表

表达式值	存储的值	屏幕的值	新的值
13	0 0 1 1	0 1 0 1	1 1 0 1
14	0 0 1 1	0 1 0 1	1 1 1 0
15	0 0 1 1	0 1 0 1	1 1 1 1

7.5 GET 语 句

True BASIC 有三个用 GET 开头的语句. 第一个可以从键盘上获得一个键的代码，第二个可以在屏幕上获得一个点的图形坐标；第三个可以获得鼠标定位器的位置和状态.

7.5.1 GET KEY 语句

一般形式:

GET KEY x

其中 x为数值变量.

执行它，等待一个键被按下，然后把被按下键的代码赋给 x. 代码为扩充的 ASCII 代码.

另外，True BASIC 还引入了键盘输入测试条件:

key input

在程序运行过程中，当键盘的某个键被按过时，则测试条件的值为真，否则为假.

把键盘测试条件 key input 和 GET KEY 语句配合使用，可以在循环中设置键盘控制. 下面以时钟程序 ex05_10 为例，增加一个 ctrl -e 控制，可编程序如下:

```
PROGRAM ex07_11
! clock
INPUT prompt "hour?":hour
INPUT prompt "minute?":minute
LET minuteo=minute
INPUT prompt "second?":second
DO
    IF key input then
        GET KEY x            ! ctrl-e code is 5
```

```
10      IF x=5 then EXIT DO
11    END IF
12    SELECT CASE second
13    CASE 0 to 58
14          LET second=second+1
15          PAUSE 0.55
16    CASE 59
17          LET second=0
18          SELECT CASE minute
19          CASE 0 to 58
20                LET minute=minute+1
21          CASE 59
22                LET minute=0
23                SELECT CASE hour
24                CASE 0 to 22
25                      LET hour=hour+1
26                CASE 23
27                      LET hour=0
28                CASE else
29                END SELECT
30          CASE else
31          END  SELECT
32     CASE else
33     END SELECT
34     PRINT using '##:##:##':hour, minute, second
35 LOOP
36 END
```

这个程序的第 8～11 行的作用是，每循环一次检查一次键盘，若按了某个键，则把该键代码赋给 x，若 x=5，则转出 DO 循环至 END 结束，若 x≠5，则不执行出口语句．若没有按任何键，则第 9～10 行不执行．因此，引入 key input 判断之后，可以消除由 GET KEY 语句引起等待按键的束缚．

7.5.2 GET POINT 语句

一般形式：

GET POINT: x,y

其中，x,y 是一点的图形坐标．

执行它，在屏幕中会自动出现光标(如十字或箭头)，这时用户可以用光标移动键(或光笔、鼠标定位器)移动光标到适当位置，再按一下回车键，这时光标当前位置的图形坐标就赋给了 x,y．注意，用户不能移动光标到窗口之外的任何一点．

例 试编一个手工画图程序．

解 这里让程序具有画点、画面、画框、画图和着色功能,程序如下:

```
PROGRAM cx07_12
!
SET window -14, 14, -10, 10
DO
   PRINT "ctrl+B begin,ctrl+E (end)"
   FOR i=-14 to 14 step 2
       LET x=i
       PLOT x, -10; x, 6
   NEXT i
   FOR j=-10 to 6 step 2
       PLOT -14, j; 14, j
   NEXT j
   GET KEY e
   IF e=2 then                         !ctrl+B
      PRINT "ctrl+Dot, Area, boxLines, Circle, coloR, Quit"
      DO
         GET KEY y
         SELECT CASE y
         CASE 4                        !ctrl+D
              GET POINT:x1, y1
              BOX AREA x1, x1+0.5, y1, y1+0.5
         CASE 1                        !ctrl+A
                 GET POINT:x1, y1
                 PLOT x1, y1
                 GET POINT:x2, y2
                 PLOT x2, y2
                 BOX AREA x1, x2, y1, y2
         CASE 12                                   !ctrl+L
                 GET POINT:x1, y1
                 PLOT x1, y1
                 GET POINT:x2, y2
                 PLOT x2, y2
                 BOX LINES x1, x2, y1, y2
         CASE 3                                    !ctrl+C
                 GET POINT:x1, y1
                 GET POINT:x2, y2
                 BOX CIRCLE x1, x2, y1, y2
         CASE 18                                   !ctrl+R
                 GET POINT x, y
                 INPUT R
```

```
                SET color R
                FLOOD x, y
            CASE 17                               !ctrl+Q
                CLEAR
                EXIT DO
            CASE else
              END SELECT
            LOOP
        ELSE IF e=5 THEN                      !crtl+E
            PRINT 'END'
            EXIT DO
        END IF
LOOP
END
```

程序运行结果之一，如图7-13所示。

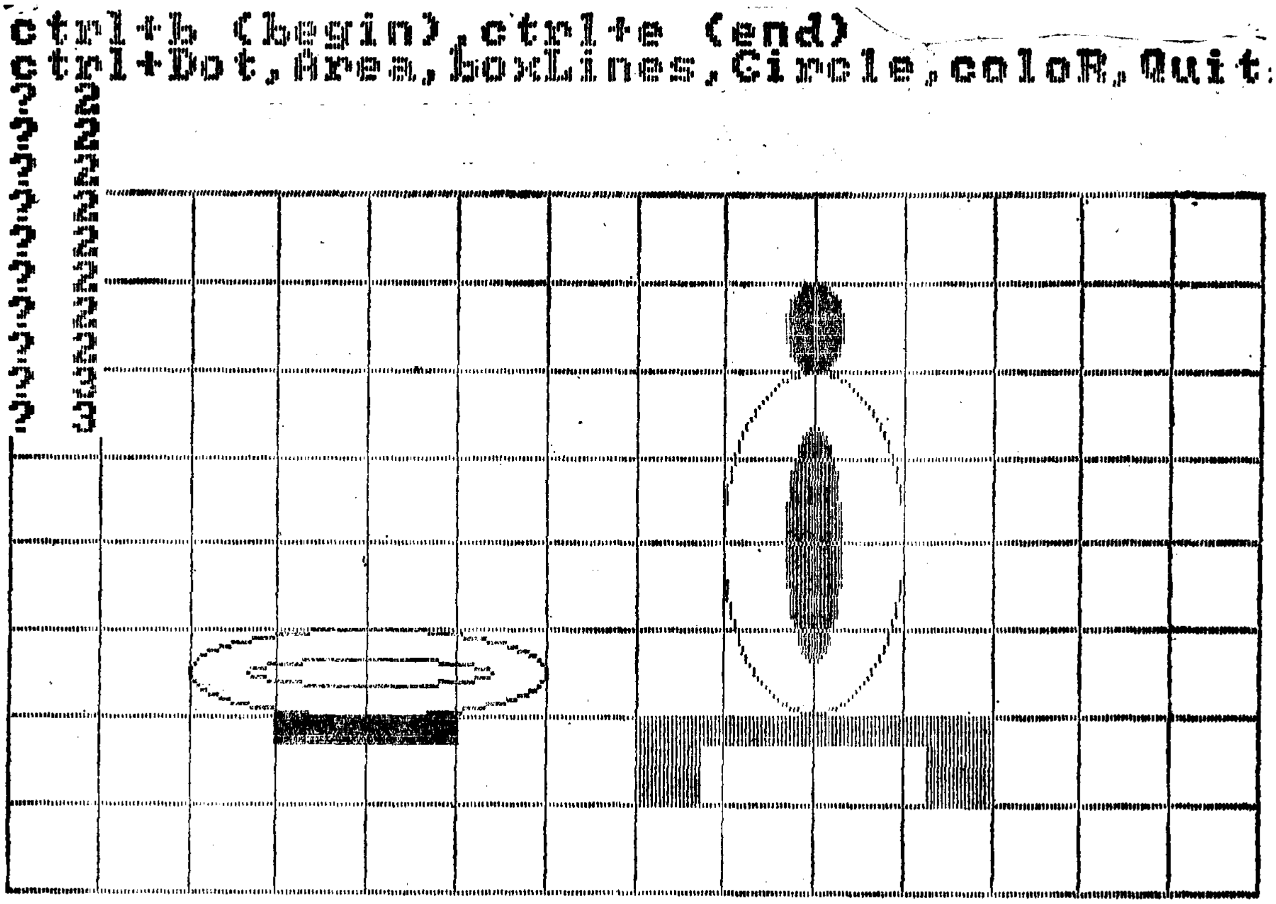


图 7-13　手工画图

7.5.3　GET MOUSE 语句

GET MOUSE 语句是 GET POINT 语句的一种扩充形式，它是特别为使用鼠标定位器而设计的.

一般形式:

GET MOUSE: x, y, state

其中,x,y 表示鼠标器在图形坐标中的位置.

state 表示鼠标当前状态,它的值含义如下:

0 鼠标定位器的键未按下.

1 鼠标定位器正被拖着走.

2 鼠标定位器的键被按下.

3 鼠标定位器的键被释放.

详细情况参看所用机器的用户指南.

7.6 字符坐标与光标的设置和询问

True BASIC 为使输入/输出更为灵活,还引入了直接设置/询问光标位置语句:

SET CURSOR line, column

ASK CURSOR vline, vcolumn

其中, line, column 是数值表达式.

vline, vcolumn 是数值变量. 参数值表示在字符坐标系上光标的行和列.

True BASIC 规定:

字符坐标系如图 7-14 所示.

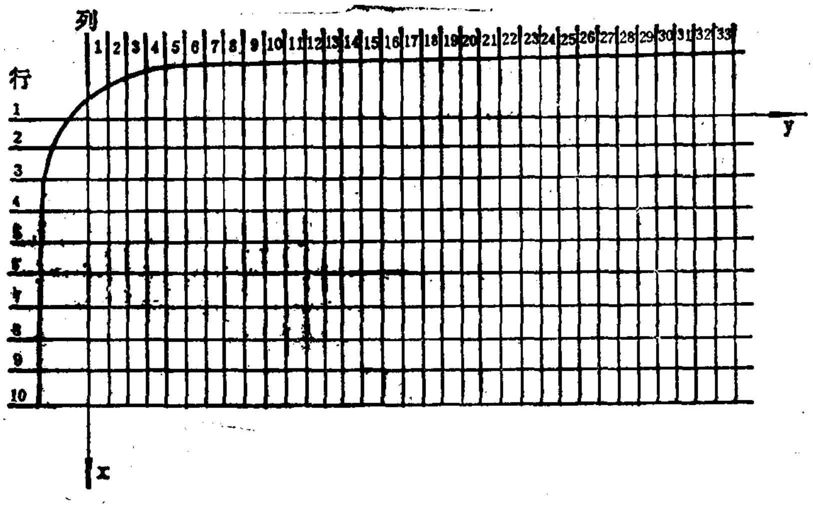


图 7-14 字符坐标系

另外, True BASIC 还引入了询问窗口最大行和最大列的语句:

ASK MAX CURSOR vline, vcolumn

其中 vline, vcolumn 为数值变量.

执行它，系统自动把行和列的最大值分别赋给 vline，vcolumn．在不同的窗口或模式下，这些值可能不同．如果你试图在允许的行列之外设置光标，将会出现运行错误信息．

例如:

```
PROGRAM ex07_13
REM input print and set cursor
!
ASK max cursor line, column
PRINT line, column
SET cursor 3, 2
PRINT "Give me two number …"
PRINT
SET cursor 7, 16
INPUT a
SET cursor 9, 16
INPUT b
ASK cursor n1, n2
SET cursor n1+2, n2+5
PRINT "The sum is";a+b
GET KEY x
END
```

运行结果:

```
Give me two number …
       ? 12
       ? 45
               The sum is 57
```

通常光标是处于发光的显示状态，为使光标不仅能处于显示状态，也能处于隐蔽状态，即不发光状态，和背景颜色一致．True BASIC 引入了光标开关语句:

SET CURSOR expr$

其中 expr$ 的值是“OFF”或“ON”．“OFF”表示光标处于隐蔽状态；“ON”表示光标处于显示状态，或分别称为打开或关闭．这在编游戏程序时很有用．

习　题

7.1 画一条对数螺线．

［提示］ $p=ae^{k\varphi}$ （其中 a，k 为常数）．

7.2 画一条圆的渐开线（用数组）．

［提示］ $\begin{cases} x=a(\cos\varphi+\varphi\sin\varphi) \\ y=a(\sin\varphi-\varphi\cos\varphi) \end{cases}$

其中 a 为圆的半径．

7.3 画各种玫瑰线，并添上不同颜色．

[提示] $\rho=a\sin b\varphi$ 或 $\rho=a\cos b\varphi$

其中 $b=2,3,4,\frac{1}{2},\frac{1}{3},\frac{3}{4}$；$a>1$.

7.4 函数

$$f(x)=\begin{cases}\frac{\pi}{2} & 0<x<\pi \\ 0 & x=0,\pm\pi \\ -\frac{\pi}{2} & -\pi<x<0\end{cases}$$

的傅立叶级数为

$$f(x)\sim 2\sum_{n=1}^{\infty}\frac{\sin(2n-1)x}{2n-1}$$

设

$$S_m(x)=2\sum_{n=1}^{m}\frac{\sin(2n-1)x}{2n-1}$$

编写一个程序，画出 $y=S_m(x)$ 在 $m=1,2,3$ 时的图形.

7.5 编写一个程序，能让一个小球作上、下跳动.

7.6 画一组逐渐放大的椭圆.

7.7 编写一个程序能用键盘输入法，在屏幕上画一幅美丽、生动的图画.

第8章　辅程序与库

8.1　辅程序概述

前面六章讲述了顺序、选择、循环三种结构的程序设计，编写了许多科学计算、事务处理和画图的程序　从程序结构的角度来说，它们都是由一个主程序组成．若是一个大的任务，还是用一个程序块来完成，势必使得程序过长．这样，既不便于编写，也不便于阅读和修改．为此，True BASIC 引入了辅程序，目的是把各个相对独立的或多次执行的操作任务，编写成一个一个的小程序块(称为辅程序)，而把主程序作为完成整个任务的控制程序．这样可使程序结构清晰，易编写、易阅读、易修改．本章将分别讲述三种辅程序的设计和应用．各个程序将是由一个主程序和一个(或几个)辅程序组成的．

8.1.1　辅程序的种类

True BASIC 辅程序分为：

函数辅程序：它又分为内部函数和外部函数．

子程序辅程序：它又分为内部子程序和外部子程序．

图画辅程序：它又分为内部图画和外部图画．

辅程序放在主程序的结束语句之后或库(辅程序集合的文件)之中，叫外部辅程序；辅程序放在主程序或外部辅程序之内叫内部辅程序．注意：True BASIC 不允许在内部辅程序之内再放辅程序，即最多嵌套一次，不准有更多重嵌套．这就使 True BASIC 既有较强的模块化功能，又使结构十分清晰．

8.1.2　辅程序的特性

辅程序是一个独立的程序单位，具体表现如下：

(1) 形式独立．各辅程序都有自己的程序模式，各自以特定语句开始和结束．

(2) 编译独立．可单独编译．

(3) 位置独立．辅程序在规定的范围内可以任意放置．

8.1.3　辅程序的执行

True BASIC 的辅程序都是可执行的程序模块．整个程序的执行是从主程序开始的，而辅程序是通过主程序的调用而执行的．如图 8-1 所示．

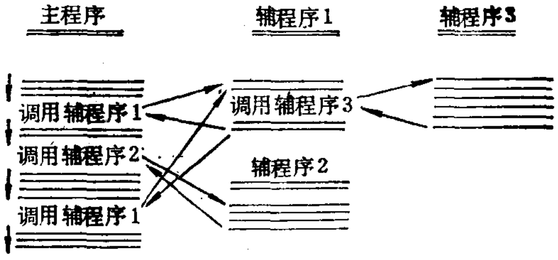


图 8-1　调用辅程序示意图

从主程序开始，程序沿箭头顺序执行，当“调用辅程序 1 ”时，转去执行辅程序 1，当程序执行

到“调用辅程序 3 ”时，再转去执行辅程序 3．辅程序 3 执行完后，返回辅程序 1 继续执行至完，然后再返回主程序继续执行．当程序执行到“调用辅程序 2 ”时，再去执行辅程序 2，如此等等．直到程序执行结束．

8.1.4 引入辅程序的优点

（1）便于编写，便于维护．

（2）实现了软件共享，辅程序是软件交流的重要形式．

（3）为软件工程化创造了条件．引入辅程序，可把一个大程序分成若干小的模块，为大型程序进行分层设计创造了条件．从软件工程的观点看，每个模块最好 50～60 行，一般不超过 200 行．但是，为了讲述语法的方便，本书所举的辅程序常常只有几条语句．

8.2 子程序的定义与调用

8.2.1 子程序定义的一般形式

```
SUB subr(fp1,fp2,…,fpi,…)
    ……}
    ……}程序体
    ……}
END SUB
```

其中 subr 为子程序名，用符号名表示，不准带 $．fpi 叫虚拟参数（简称虚参），用符号名表示．通过参数可以建立程序块之间的数据联系．参数设置根据程序员的需要可任意选择．若不选，则括号也就不要了，称为无参子程序．

程序体和主程序体设计一样，根据需要可使用任何语句（特殊声明的专用语句除外）．此外，在子程序体内可使用如下语句：

```
EXIT SUB
```

执行它，使控制直接返回到调用它的语句．把子程序看作一个基本结构这是符合结构化程序设计“单口入、单口出”准则的．

8.2.2 子程序的调用

子程序的执行是通过如下调用语句实现的：

```
CALL subr(rp1,rp2,…,rpi,…)
```

其中 rpi 叫实在参数（简称实参），True BASIC 允许递归调用（详见8.6）．

8.2.3 子程序应用举例

例 1 求数组A(10)的倒置数组，且放在原来的数组中．

解 先编一个对半交换数组元素的主程序，然后，再编一个交换变量值的子程序．

```
1  PROGRAM ex08_1
```

```
SUB change(a, b)
      LET wk=a
      LET a=b          } 内部子程序
      LET b=wk
END SUB
DIM a(10)
MAT INPUT a
FOR i=1 to 5
      CALL change(a(i), a(11-i))
NEXT i
MAT PRINT a;
END
```

例 2 打印如图 8-2 所示的表格.

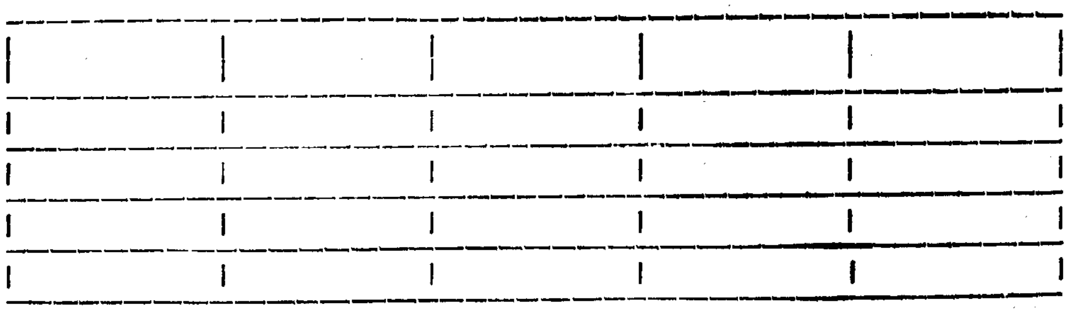


图 8-2

解 可以把此表看作是由一种横线和一种竖线组成的. 分别编两个子程序，再根据表格形式组织打印. 可编程序如下:

```
PROGRAM ex08_2
! print tabl
CALL line1
CALL line2
FOR i=1 to 5
      CALL line2
      CALL line1
NEXT i
END
SUB line1
      FOR k=1 to 5
            PRINT "----";
      NEXT k              } 外部子程序 1
      PRINT "-"
END SUB
SUB line2
```

```
      FOR k=1 to 5
          PRINT "|  ";
      NEXT k                   } 外部子程序2
      PRINT "|"
END SUB
```

8.3 函数的定义与调用

8.3.1 函数定义的一般形式

DEF func(fp1,fp2,…)=expr

或 **DEF func(fp1,fp2,…)**

```
     ......
     ......
     LET func=expr      } 函数体
     ......
     ......
END DEF
```

其中,func 为函数名,用符号名表示.函数和常数、变量一样,分两类:一类叫数值型,一类叫字符串型,用函数名是否带 $ 来区别. 函数名在函数体部分至少要被赋值一次.

fp 叫虚拟参数(简称虚参),用符号名表示.通过设置参数可以建立程序块之间的数据联系.参数设置可依据程序员的需要任意选择. 可以不选,那时,括号也可以省略.

函数体和主程序体一样,除去特殊声明的专用语句外,根据需要可用任何语句.还可用 EXIT DEF 语句,使函数直接返回到它的调用处.

注意: EXIT DEF 语句不准在函数体以外使用.

注. 美国国家标准,DEF 要用 FUNCTION 代替.

8.3.2 函数的调用

(1) 内部函数调用. 和标准函数一样. 直接用

func (rp1,rp2,…)　　　　(rp 为实在参数)

出现在表达式中.

(2) 外部函数调用. 首先要在程序开头用如下语句说明:

DECLARE DEF func1,func2,…

其中,func 为函数名. 可以一次说明多个,用逗号隔开. 如果有较多的外部函数,可以使用几个 DECLARE 语句. 然后和标准函数调用一样,直接用

func(rp1,rp2,…)　　　　(rp 为实在参数)

出现在表达式中.

True BASIC 允许递归调用(详见 8.6).

8.3.3 函数应用举例

例 1 打印一个串的逆串.

解 用内部函数，可编程序如下:

```
PROGRAM ex08_3
!
DEF reverse$(S$)
      LET result$=" "
      FOR i=len (s$) to 1 step -1
            LET resul t$=resul t$ & S$[i:i]
      NEXT i
      LET reverse$=result$
END DEF
INPUT prompt "string?":a$
PRINT reverse$(a$)
END
```

例 2 编写一个在长串中用新的子串置换旧的子串的程序.

解 用外部函数，可编程序如下:

```
PROGRAM ex08_4
! Replace all occullences of a substring
    with some other string.
DECLARE DEF replace$                    ! replace$ is an external
DO                                      !             function
     PRINT "string, old text, new text";
     INPUT s$, o$, n$
     PRINT replace$(s$, o$, n$)
LOOP until s$=" "
END
!
DEF replace$(str$, old$, new$)
      LET n=1                           ! start at start of str$
      DO                                ! loop finding metches
         LET n=pos (str$, old$, n)
         IF n=0 then EXIT DO
         LET str$[n:n+len(old$)-1]=new$
         LET n=n+len(new$)                   ! skip past new next
      LOOP
      LET replace$=str$
END DEF
```

例 3 试编一个用梯形法求 f(x) 定积分

$$\int_a^b f(x)\,dx$$

的程序，并用 $f(x)=x$, $a=0$, $b=1$ 验证之.

解 从微积分学可知，当 $x\in[a,b]$, $f(x)\geqslant 0$ 时，$f(x)$ 在 $[a,b]$ 上的定积分就是 $f(x)$ 曲线下的面积，如图 8-3 所示. 这里采用定步长梯形法. 这个方法就是把这块面积分为 n 个小梯形，然后求它们的面积和. 当 n 取得足够大时，则折线逼近曲线，积分趋于准确值.

若把 $[a,b]$ 分为 n 等分，并令 $h=\dfrac{b-a}{n}$，则

$$\int_a^b f(x)\,dx=\sum_{i=0}^{n-1}\frac{h}{2}[f(x_i)+f(x_i+1)]$$

$$=\left[\frac{1}{2}(f(a)+f(b))+\sum_{i=1}^{i} f(x_i)\right]$$

其中

$$x_i=a+ih=a+\sum_{j=1}^{i} h \quad (i=1,2,\cdots,n-1).$$

根据上述分析，可编写一个梯形法求积分的辅程序，和一个被积函数辅程序. 如果被积函数改变时，只要改一下被积函数辅程序即可.

程序 1：

```
PROGRAM ex08_5a
!
DECLARE DEF trap
INPUT a, b, n
LET total=trap(a, b, n)
PRINT n, a, b, total
END
! Trapezoidal Method
DEF trap (a, b, n)
     DECLARE DEF f
     LET h=(b—a)/n
     LET wtrap=(f(a)+f(b))/2
     LET x=a
     FOR i=1 to n—1
          LET x=x+h
          LET wtrap=wtrap+f(x)
     NEXT i
     LET trap=h*wtrap
END DEF
!
DEF f(x)
     LET f=x
END DEF
```

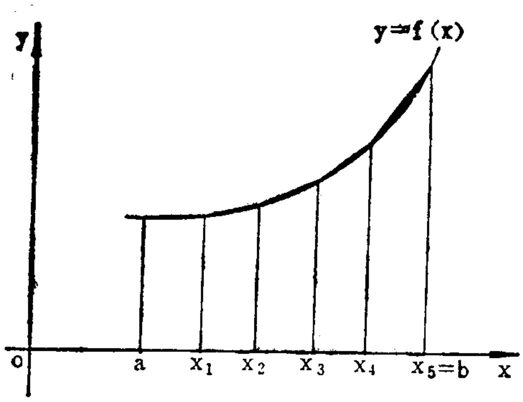


图 8-3 梯形法求积方

这是一个外部函数又调用另一个外部函数的例子.

程序 2:

```
PROGRAM ex08_5b
!
DECLARE DEF trap
INPUT a, b, n
LET total=trap(a, b, n)
PRINT n, a, b, total
END
! Trapezo i dal Method
DEF trap(a, b, n)
    !
    DEF f(x)
        LET f=x
    END DEF
    !
    LET h=(b-a)/n
    LET wtrap=(f(a)+f(b))/2
    LET x=a
    FOR i=1 to n-1
        LET x=x+h
        LET wtrap=wtrap+f(x)
    NEXT i
    LET trap=h*wtrap
END DEF
```

这是一个外部函数调用另一个内部函数的例子.

8.3.4 子程序与函数的比较

表 8-1 子程序与函数的比较

特性＼程序		函　数	子　程　序
辅程序名	名　字	用字符名表示	用符号名表示(不带$)
	类　型	名字带$或不带$	无类型
	值	有值	无值
辅程序体		函数名要被赋值	子程序名不准被赋值
		可以使用EXIT DEF语句	可以使用EXIT SUB语句
虚参个数(n)		n≥0	n≥0
调用方法		作表达式的初等量	用CALL语句
功　能		计算一个值	计算多个值,或操作

子程序与函数的比较如表 8-1 所示. 通过比较不难看出，子程序比函数具有更广泛的用途.

8.4 虚实结合

8.4.1 虚实结合概述

如前所述，为建立不同程序块之间的数据联系，在辅程序中可以设置虚参，在调用时给出实参，通过虚实结合，实现这种联系. True BASIC 规定：子程序虚参可以是变量、数组和通道号；函数虚参可以是变量、数组，而不能是通道号. 有关通道问题详见第 9 章.

虚实结合是按参数在参数表中位置对应的原则结合的，所以要求实参与虚参的个数、顺序、类型一致. 具体规定如表 8-2 所示.

表 8-2

虚参	实参	函数结合方式	子程序结合方式
变量名	常数，表达式①	值结合	值结合
	变量，下标变量	值结合	名结合
数组说明②	数组名	值结合	名结合
通道号	通道号	不准用	值结合

① 这里"表达式"不包括简单变量和下标变量.

② 这里"数组说明"形式，如 a( ),b(,)c(,,)，分别表示一维、二维、三维数组(详见 8.4.3).

8.4.2 值结合与名结合

值结合，是把实参拷贝到一个临时变量中，在辅程序中使用临时变量参加运算，结果不改变实参. 这就是说，程序实参决不会因为调用辅程序而改变. 因为实参与虚参之间是单向传递的，即

实参 ⟶ 临时变量 ⟷ 虚参

名结合，是在辅程序中直接使用实参. 这样当辅程序虚参被改变时，则实参也就改变了. 因实参与虚参之间是双向传送的，即

实参 ⟷ 虚参

True BASIC 规定：在函数调用中虚实结合都是值结合，因此，调用函数决不会改变调用程序中实参的值. 这就避免了通常所谓的"函数的副作用".

8.4.3 数组虚实结合

要注意，实数组与虚数组维数要一致.

例 1 求方阵对角线元素构成的数组，和主对角线上元素的乘积.

解 编写一个带子程序的程序:

```
PROGRAM ex08_6
DIM a(5,5), b(5)
MAT INPUT prompt "array(5,5)":a
CALL subr (d, a, b)
MAT PRINT a;b;
PRINT
END
!
SUB subr (dia, x(,), y( ))
      LET dia=1
      FOR i=lbound(y) to ubound(y)
            LET y(i)=x(i,i)
             LET dia=dia*y(i)
      NEXT i
END SUB
```

实数组与虚数组的结合,如图 8-4 所示.

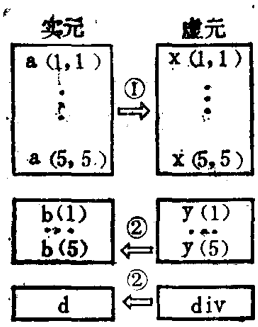
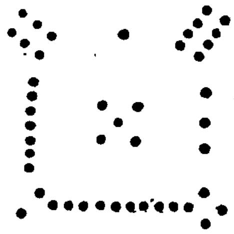


图 8-4 实元与虚元的结合

例 2 打印洛书.

河图与洛书都是在我国黄河、洛水一带发现的甲骨文化,距今已有五六千年的历史. 洛书刻有一到九的九个自然数排列图,如图 8-5 所示.



6	1	8
7	5	3
2	9	4

图 8-5 洛书

洛书的每一行、每一列以及对角线上三个数之和都是 15, 称之为洛书常数. 洛书的奇妙结构令人叹服! 故也称为"魔方". 洛书是中华民族祖先高级思维的光辉成就, "洛书是中华民族原始文化的标志."

试编写一个子程序,填写 n(奇数)阶方阵的洛书.

解 经研究发现,洛书具有如下规律:

(1) 数值从 1 到 n^2,依次填入.

(2) 数值 1 填入第 1 行正中一列的位置.

(3) 后继数填入前一个数所在位置的行数减 1 列数减 1 的位置.

若行数出阵,列数不出阵,则行数取值为 n,列数不变(如图 8-5 的数字 2).

若行数不出阵,列数出阵,则行数不变,列数取值为n(如图 8-5 的数字 3).

(4) 每当连续填入 n 个数后,则行数加 1,列数不变.

根据上述算法，选择数组 a 模拟 n 行乘 n 列方阵，i，j 表示数组 a 的任一元素的行和列，k 表示填入数组 a 的自然数 1 到 n^2. 按上述第(4)条规律，n 阶方阵要有 n 个周期，设 l 表示周期数，m 表示每个周期填入数的个数，从 1 到 n. 根据上述分析可编程序如下：

```
PROGRAM ex08_7
!
DIM x(7,7)
CALL mofang(7,x)
MAT PRINT using "##":x
END
SUB mofang(n, a(,))
      LET k=0
      LET i=1
      LET j=int(n/2) +1
      FOR l=1 to n
            LET k=k+1
            LET a(i,j)=k
            FOR m=2 to n
                  LET i=i-1
                  LET j=j-1
                  IF i<=0 then LET i=n
                  IF j<=0 then LET j=n
                  LET k=k+1
                  LET a(i,j)=k
            NEXT m
            LET i=i+1
      NEXT l
END SUB
```

运行结果：

28	19	10	1	48	39	30
29	27	18	9	7	47	38
37	35	26	17	8	6	46
45	36	34	25	16	14	5
4	44	42	33	24	15	13
12	3	43	41	32	23	21
20	11	2	49	40	31	22

8.5 全局变量与局部变量

如前所述，内部辅程序可以嵌套在主程序或外部辅程序之中. 这样，同一个变量名可以出现

在主程序或外部辅程序之中，也可以出现在内部辅程序之中，那么，它们到底有没有关系呢？是否可以视为同一个变量呢？为此 True BASIC 引入了全局变量与局部变量的概念.

全局变量指的是主程序与其内部辅程序，或外部辅程序与其内部辅程序公用的变量. 除虚拟参数之外，在内部辅程序中，和外层同名的变量都是全局变量.

局部变量指的是仅用于某个辅程序的变量. 对主程序来说，在外部辅程序中的所有变量都是局部变量，除去虚实结合关系之外，即使变量名相同也互不相干.

外部辅程序与内部辅程序在可读性、运行效率等各方面各有利弊，为减少全局变量带来的干扰，建议初学者最好多用外部辅程序，它有很强的独立性. True BASIC 的特点是：不允许内部辅程序再有自己的内部辅程序. 这样只允许一次嵌套，哪些是全局变量，哪些是局部变量，也比较易于观察和分析，只要细心些，错误也是可以避免的.

例 1 阅读如下两个程序，并给出结果：

程序 1：

```
PROGRAM ex08_8a
!
LET a, b=3
CALL pro(a, b)
CALL pro(a, b)
PRINT "m=";m, "n=";n
END
SUB pro(x, y)
      LET m=x*y
      LET x=x+5
      LET y=y+5
      LET n=x*y
      PRINT "x=";x, "y=";y
END SUB
```

运算结果：

```
x=8        y=8
x=13       y=13
m=0        n=0
```

虽然变量 m, n 在辅程序运行中曾两次被赋值，因为它是外部辅程序的局部变量，与主程序无关. 所以在主程序中，打印它们，其值仍为 0.

程序 2：

```
PROGRAM ex08_8b
!
SUB pro(x, y)
      LET m=x*y
      LET x=x+5
```

```
        LET y=y+5
        LET n=x*y
        PRINT "x=";x,"y=";y
END SUB
!
LET a,b=3
CALL pro(a,b)
CALL pro(a,b)
PRINT "m=";m,"n=";n
END
```

运算结果:

```
x=8     y=8
x=13    y=13
m=64    n=169
```

虽然变量 m,n 在主程序中未被赋值，而在辅程序中曾两次被赋值. 由于 m,n 是全局变量，所以在主程序中打印它，其值不是 0 .

例 2　已知 n 组观测值 (x_i,y_i) $(i=1,2,3,\cdots,n)$，用最小二乘法，求最佳直线 $\hat{y}=a+bx$，如图 8-6 所示.

解　如果变量 x 和 y 之间存在着线性关系，则可用直线

$$y=a+bx$$

来拟合它们之间的变化关系. 最小二乘法就是让 a, b 满足

$$\sum_{i=1}^{n}[y_i-\hat{y}_i]^2=\sum_{i=1}^{n}[y_i-(a+bx_i)]^2=\text{最小值}$$

由此导出

$$\begin{cases} b=\dfrac{\sum\limits_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sum\limits_{i=1}^{n}(x_i-\overline{x})^2} \\ a=\overline{y}-b\overline{x} \end{cases}$$

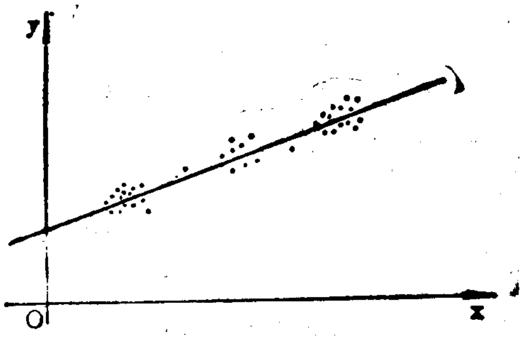


图 8-6　最佳直线图

其中

$$\overline{x}=\frac{1}{n}\sum_{i=1}^{n}x_i,\ \overline{y}=\frac{1}{n}\sum_{i=1}^{n}y_i$$

方程 $\hat{y}=a+bx$ 叫回归方程. 当任意给定一个 x 值时，可利用它预报 y 的值. 依据上述分析可编程序如下:

程序 1:

```
PROGRAM ex08_9a
!
DIM x(100),y(100)
```

```
INPUT prompt "n?":n
FOR i=1 to n
     INPUT x(i),y(i),
NEXT i
PRINT
CALL lsq(p,q)
PRINT "a=";p,"b=";q
FOR  i=1 to n
       LET e=y(i)-(p+q*x(i))
       PRINT i,x(i),y(i),e
NEXT i
SUB  lsq  (a,b)
      LET sx,sy=0
      FOR i=1 to n
           LET sx=sx+x(i)
           LET sy=sy+y(i)
      NEXT i
      LET xbar=sx/n
      LET ybar=sy/n
      LET sx,sy=0
      FOR i=1 to n
           LET sx=sx+(x(i)-xbar)^2
           LET sy=sy+(x(i)-xbar)*(y(i)-ybar)
      NEXT i
      LET b=sy/sx
      LET a=ybar-b*xbar
END SUB
END
```

因为x,y和n是全局变量,所以不必列为内部子程序的虚拟参数.

程序2:

```
PROGRAM ex08_9b
!
DIM x(100),y(100)
INPUT prompt "n?":n
FOR i=1 to n
     INPUT x(i),y(i),
NEXT i
PRINT
CALL lsq(x,y,n,a,b)
PRINT "a=";a,"b=";b
FOR i=1 to n
```

```
        LET e=y(i)-(a+b*x(i))
        PRINT i, x(i), y(i), e
NEXT i
END
SUB lsq(x( ), y( ), n, a, b)
        LET sx, sy=0
        FOR i=1 to n
              LET sx=sx+x(i)
              LET sy=sy+y(i)
        NEXTi
        LET xbar=sx/n
        LET ybar=sy/n
        LET sx, sy=0
        FOR i=1 to n
              LET sx=sx+(x(i)-xbar)^2
              LET sy=sy+(x(i)-xbar)*(y(i)-ybar)
        NEXT i
        LET b=sy/sx
        LET a=ybar-b*xbar
END SUB
```

因为x,y和n是局部变量,所以,必须列为外部子程序的虚拟参数。

运行结果:

```
n?10
? 2, 2.1, 3.1, 3, 2.5, 2.52, 8.1, 8.22,
? 9, 9.05, 6.7, 6.5, 7.9, 7.89, 10.1, 10.2,
? 4, 4.3, 5, 5.4
a= .109019              b= .994689
1                        2          2.1          1.60425e-3
2                        3.1        3            -.192553
3                        2.5        2.52         -7.57401e-2
4                        8.1        8.22         5.40038e-2
5                        9          9.05         -.011216
6                        6.7        6.5          -.273432
7                        7.9        7.89         -7.70585e-2
8                        10.1       10.2         4.46265e-2
9                        4          4.3          .212227
10                       5          5.4          .317538
```

8.6 递　　归

True BASIC 规定:

(1) 若两个辅程序是前后并列的,则可以相互调用(ex08_13).

(2) 若两个辅程序是内外嵌套的,则

a) 外层可以调用内层(ex08_12).

b) 内层可以调用外层.

(3) 一个辅程序自己可以调用自己(ex08_10,ex08_11).

辅程序自己调用自己通常称为递归调用，其中直接地自己调用自己叫直接递归；间接地自己调用自己叫间接递归.

8.6.1　直接递归

例1　求n!.

解　数学上阶乘定义为

$$0!=1$$

$$n!=n\cdot(n-1)!(若\ n>0)$$

这是一个递归定义,可把它编写成如下递归函数辅程序:

```
PROGRAM ex08_10
DEF f(n)
      IF n>0 then
         LET f=n*f(n-1)          (返回点 1 )
      ELSE
          LET f=1
      END IF
END DEF
!
PRINT "input num:"
INPUT num
LET fofn=f(num)              (返回点 2 )
PRINT "The factorial of";num;"is";fofn
END
```

递归调用通常导致对它若干次调用,如 num=3, 将导致对函数 f 4 次调用. 分配存储情况如图 8-7 和图 8-8 所示.

从此例看出,在每次调用时,系统程序都要为函数 f 的参数分配存储单元，用来存储它的实元和返回点,以便返回后,程序从此处继续执行. 显然,执行递归过程,既费时间,又费存储空间. 但是，由于递归与自然表述更接近，具有易理解、易设计、结构清晰的优点，故递归法是一种常用的方法.

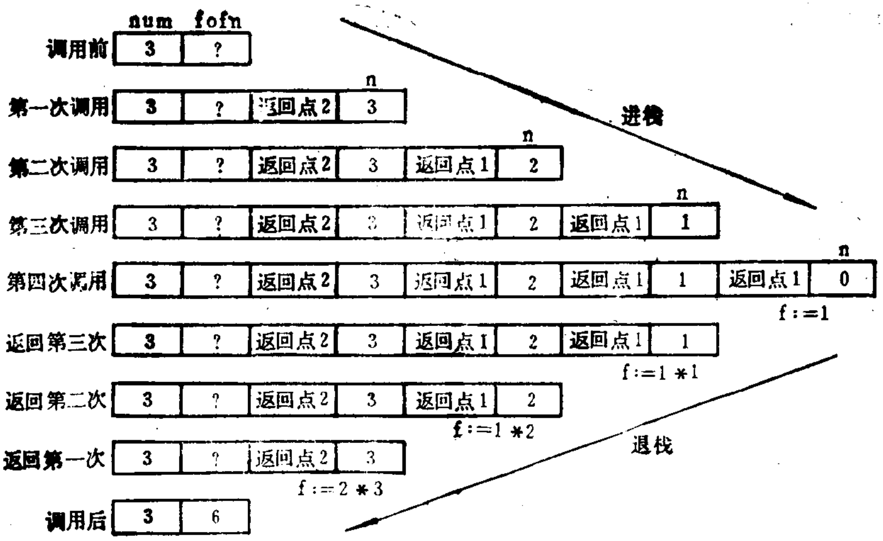


图 8- 求 3! 进栈退栈图示意图

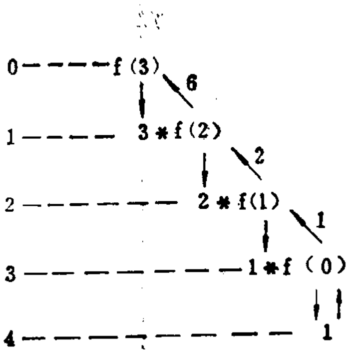


图 8-8 求 3! 递归跟踪图

例 2 输入一个数(如 125),把它按倒序输出来(如 521).

解 可编程序如下:

```
PROGRAM ex08_1
SUB RevD(number)
    PRINT chr$(mod(number, 10)+48);
    LET number=int(number/10)
    IF number  > 0 then CALL RevD(NUMBER)
END SUB
PRINT "please enter a positive integer:";
INPUT numbe
CALL RevD(number)
END
```

概括地说,递归法求解分为如下两步:第一步是把问题变换成一个与原来问题等价,但某些方面更简单的新问题,如把 n! 化为 n*(n-1)!. 第二步重复作这个变换,直到这个问题变得很简单. 如最后求 0!,数学定义为 1.

例 3 汉诺塔(Tower of Hanoi)趣味程序.

约在十九世纪末,欧洲珍奇商店出售一种称为汉诺塔的玩具,这种玩具是由 8 个中间有孔的纸盘和一个装有三根木针的木架组成,如图 8-9 所示. 游戏的目标是把左边针上的塔移到右边的针上,条件是一次只能移动一个盘,并且不允许大盘放在小盘上面.

假定木针位置分别记作 s1, s2, s3,圆盘个数为 n. 任务是把 n 个盘从 s1 移到 s3 上,我们用子程序

mtower(n, s1, s3, s2)

表示这个任务,这里第一个参数表示要移动的圆盘的个数,第二个参数表示取出的位置,第三个

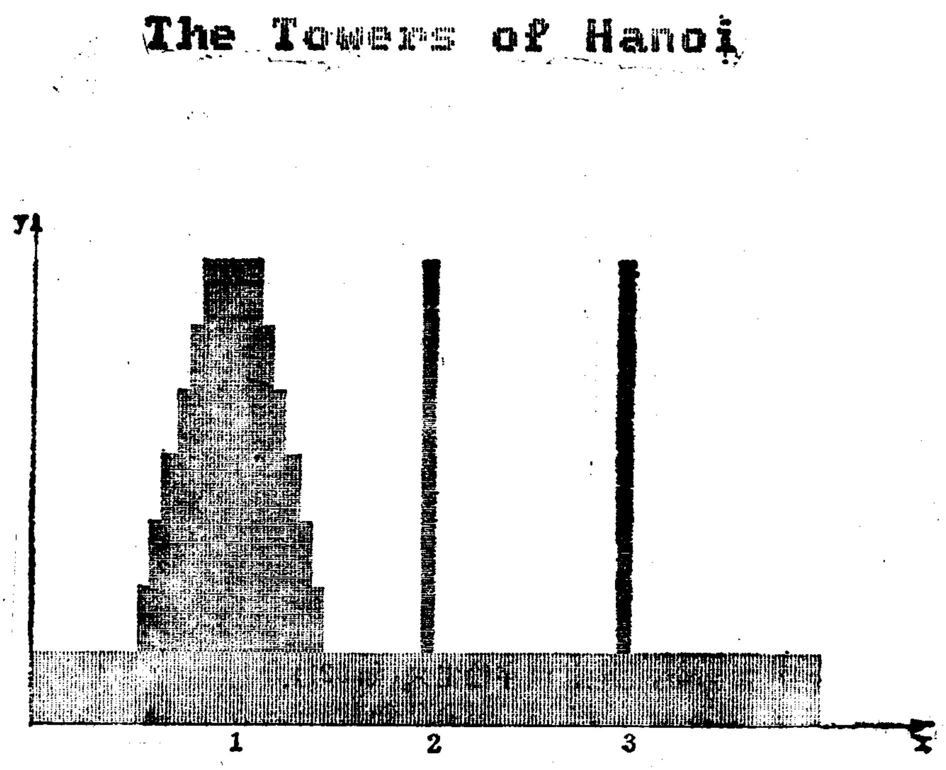


图 8-9 汉诺(Hanoi)塔

参数表示放入位置,第四个参数为临时位置.

让我们把着眼点放在最大圆盘上,要实现最大盘从 s1 移动到 s3 上,则必须完成如下任务:

mtower(n−1,s1,s2,s3)

即把 n−1 个盘从 s1 移到 s2 上. 这时用

mdisk(s1,s3)

表示从 s1 取一个盘移动到 s3 上. mdisk 子程序的第一个参数表示取出的位置, 第二个参数表示放入的位置. 如若再完成

mtower(n−1,s2,s3,s1)

即把 n−1 个盘从 s2 移到 s3 上,任务到此完成.

虽然,还留下两个移动 n−1 个盘的任务未完成, 但问题已朝着解决的方向前进了关键的一步,重复下去,直到要移动盘的个数为 1 时, 无疑是可以完成的. 依据上述分析,用递归法可编程序如下:

```
PROGRAM ex08_12
!
INPUT prompt "total":total
CALL mtower(total, 1, 3, 2)
END
!
SUB mtower(n, s1, s3, s2)
      SUB mdisk(off, put)
```

```
            PRINT "form";off;"move to ";put
        END SUB
        IF n > 0 then
           CALL mtower(n-1, s1, s2, s3)
           CALL mdisk(s1, s3)
           CALL mtower(n-1, s2, s3, s1)
        END IF
  END SUB
```

这是外部子程序递归调用的例子，同时它还调用了内部子程序.

运行结果(n=3):

```
total3
   from 1 move to 3
   from 1 move to 2
   from 3 move to 2
   from 1 move to 3
   from 2 move to 1
   from 2 move to 3
   from 1 move to 3
```

这个程序的运行结果，给出了一张移动盘的操作过程表，按照表顺序地去移动即可.

下面的程序 ex08_12a 是用动画的形式表示汉诺塔游戏(n=6)，这是一个较复杂的程序，供读者阅读.

```
PROGRAM ex08_12a
   ! Towers of Hanoi
   !
   LET disknum=6
   CALL initial(disknum)
   !
   DIM xsize(3, 0 to 6)          ! For graphics
   LET xsize(1, 0)=disknum       ! All on disk 1
   FOR y=1 to disknum
       LET xsize(1, y) = disknum+1-y
   NEXT y

   CALL mtower(disknum, 1, 3, 2, xsize)
 END

 SUB initial(n)

       SET WINDOW 0, 4, 0, 8
       SET BACK 17
```

```
        SET COLOR 1
        PLOT TEXT, at .5, 10: "The Towers of Hanoi"

        SET COLOR 1
        BOX AREA 0, 4, 0, 1
        SET COLOR 3
        FOR x = 1 to 3
            BOX AREA x-.02, x+.02, 1.05, 7
        NEXT x
        FOR y = 1 to n
            CALL disk(1, y, 2, n+1-y)
        NEXT y

END SUB

SUB mtower(n, s1, s3, s2, xsize(,))          ! Recursive routine
        IF n = 1 then
           CALL mdisk(s1, s3, xsize)
        ELSE
           CALL mtower(n-1, s1, s2, s3, xsize)
           CALL mdisk(s1, s3, xsize)
           CALL mtower(n-1, s2, s3, s1, xsize)
        END IF
END SUB

SUB mdisk(off, put, xsize(,))
      LET offp = xsize(off, 0)
      LET putp = xsize(put, 0)
      LET    xs = xsize(off, offp)
      CALL disk(off, offp, 0, xs)                 ! Erase
      CALL disk(put, putp+1, 2, xs)               ! Draw
      LET    xsize(off, 0) = offp-1
      LET    xsize(put, 0) = putp+1
      LET xsize(put, putp+1) = xs
END SUB
SUB  disk  (x, y, col, xs)             ! Disk no., height, color, size
       LET x1=x-(xs+1)/15
       LET x2=x+(xs+1)/15
       LET y1=y
       LET y2=1+y
       SET COLOR col
```

```
        BOX AREA x1, x2, y1+.05, y2
        IF col=0 then
           SET COLOR 3
           BOX AREA x-.02, x+.02, y1+.05, y2
        END IF
        PAUSE .2
   END SUB
```

这个程序由一个主程序和四个子程序组成. initial 子程序画出了汉诺塔游戏的初始状态. mtower 子程序和 ex08_12 程序的 mtower 子程序功能一样. mdisk 子程序功能是移动盘,即抹去一个盘,再画一个盘. disk 子程序是画盘的子程序. 为表示各个时刻圆盘在针上的分布,引入了 xsize 数组,它的第 1 个下标表示木针,第 2 个下标表示塔高,数组元素的值表示盘的大小(即宽度) 而 xsize(1,0),xsize(2,0), xsize(3,0)表示木针上盘的个数. 运行过程画面之一,如图 8-10 所示.

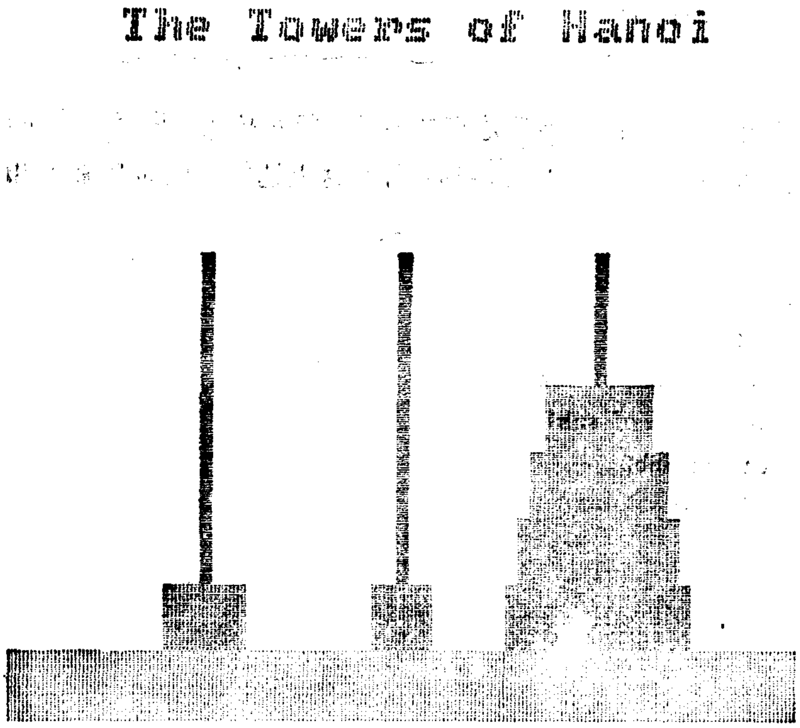


图 8-10 汉诺塔移动中间画面

8.6.2 间接递归

间接递归即间接地自己调用自己,请看如下程序:

```
PROGRAM ex08_13
SUB subl(fbl)
     LET n=n+1
     PRINT tab(4*n) ;n;' begin of subl'
     IF fbi=1 then CALL sub2
     PRINT tab(4*n);n;' end of subl'
```

```
        LET n=n-1
END SUB
SUB sub2
        LET n=n+1
        PRINT tab(4*n);n;' begin of sub2'
        LET b1=0
        CALL sub1(b1)
        PRINT tab(4*n);n;' end of sub2'
        LET n=n-1
END SUB
!
LET n=0
LET b1=1
CALL sub1(b1)
PRINT 'end of ex08_13'
END
```

这个程序有两个并列的子程序，而且是相互调用，即间接递归调用．当sub1子程序的参数fb1 $\neq 1$时，递归调用结束．为说明递归调用的层次，这里选择了全局变量n，用n表示程序递归过程的路径，运行结果如下：

```
    1    begin of sub1
         2    begin of sub2
              3    begin of sub1
              3    end of sub1
         2    end of sub2
    1    end  of sub1
end of ex08_13
```

8.7 图画的定义与调用

和科学计算、数据处理一样，画图也具有独立性、重复性和共享的需求，为此True BASIC引入了“图画”．图画和子程序一样是True BASIC的一种程序模块，它可以有虚参，通过虚实结合可以传送数据．图画也分内部图画和外部图画等，这些和子程序完全类似．不同地是对图画可以作“变换”，使图画产生千姿百态的变化．

8.7.1 图画定义的一般形式

PICTURE pic(fp1,fp2,…)

```
    ……
    PLOT
    ……      }程序体
    ……
END PICTURE
```

其中 pic 为图画辅程序名，用符号名表示，不带$.

fp 为虚拟参数（简称虚参），用符号名表示. 虚参依据程序员的需要可任意选择. 若不选虚参，则括号也就不必出现了.

程序体和 True BASIC 主程序体一样，可使用各种语法成分，但程序体内一定要包含PLOT等画图语句，此外也可以使用 EXIT PICTURE 语句使控制直接返回调用程序. 图画调用有三种形式，下面分三小节给予叙述.

8.7.2 图画调用形式之一

DRAW pic(rp1,rp2,…)

其中 rp 为实在参数（简称实参），用符号名表示.

例 1 画一系列由小到大的矩形框.

解 用 PLOT 语句编写一个画矩形框的图画程序，然后再多次调用它，可编程序如下：

```
PROGRAM ex08_15
! Make a "square" picture, and draw it.
!
PICTURE square(size)
        PLOT -size, size;
        PLOT size, size;
```

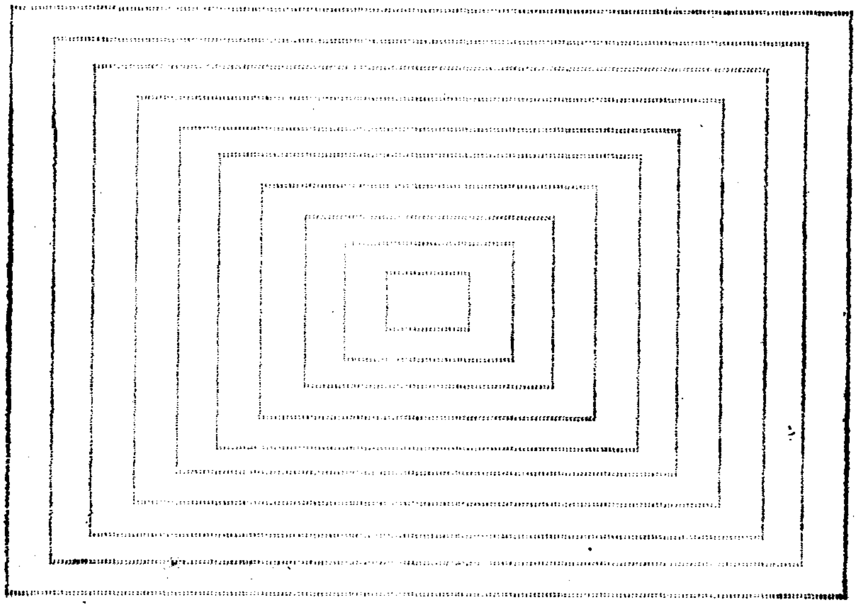


图 8-11

```
        PLOT   size, -size;
        PLOT  -size, -size;
        PLOT  -size,  size
END PICTURE
SET window -15, 15, -15, 15
FOR i=1 to 10
     DRAW square(i)
NEXT i
END
```

运行结果，如图 8-11 所示.

这个程序也可用 BOX 语句编写，程序更简单.

例 2 正多边形和圆的关系.

解 本题将用画图法演示，当正多边形的边逐步增加时，最后将成为一个圆. 极坐标与直角坐标关系如下:

$$\begin{cases} x = r\cos(\varphi) \\ y = r\sin(\varphi) \end{cases}$$

依据上述分析，编写一个画任意正多边形的图画程序，然后在主程序中多次调用它，程序如下:

```
1   PROGRAM ex08_16
2   !
3   SET window -14, 14, -10, 10
4   ASK max color cmax
5   FOR j=3 to 25
6        SET color mod(j, cmax)
7        DRAW polygom(j)
8   NEXT j
9   END
10  !
11  PICTURE polygom (sides)
12      LET alpha=2*pi/sides
13      FOR i=0 to sides
14          LET u=i*alpha
15          PLOT 10*cos(u), 10*sin (u);
16      NEXT i
17      PAUSE 1
18      PLOT
19  END PICTURE
```

程序的第 18 行作用是结束一个正多边形笔画. 没有这个语句，各多边形之间将有一条连线. 程序运行结果如图 8-12 所示.

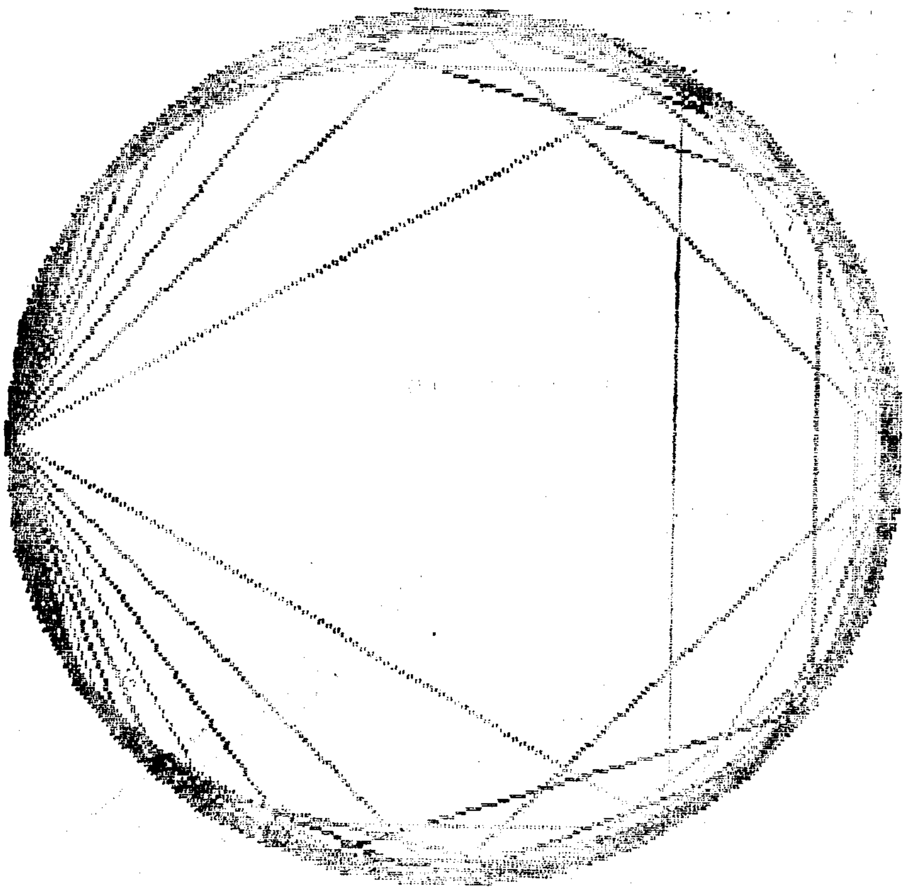


图 8-12

8.7.3 图画调用形式之二

DRAW pic(rp1,rp2,…) with trans

其中 DRAW 和 with 为关键字.

trans 为图画变换,即是一个对图画大小或形状调整的操作. True BASIC 提供了如下五种内部变换:

shift(a,b) 平移整个图画,点(x,y)将放在(x+a,y+b)处.

scale(a) 改变图画大小,点(x,y)将放在(a*x,a*y)处.

scale(a,b) 改变图画大小,点(x,y)将放在(a*x,b*y)处.

rotate(α) 旋转图画 α弧度(或角),以图形坐标原点为中心,沿顺时针方向旋转.

shear(α) 倾斜图画 α弧度(或角),将图画中所有垂直线,沿顺时针方向倾斜.

注意: 只能对用 PLOT 语句画的图画作变换,而对用 BOX 语句画的图画无效.

例 1 对矩形作旋转变换.

解 可编程序如下:

```
1  PROGRAM ex08.17
2  ! Make a "square" picture, and draw it.
```

```
!
PICTURE square(size)
      PLOT  -size, size;
      PLOT   size, size;
      PLOT   size, -size;
      PLOT  -size, -size;
      PLOT  -size,  size;
END PICTURE
SET  window  -15, 15, -15, 15
FOR  i=1  to  10
       DRAW square(i)  with  rotate(i*pi/40)
NEXT i
END
```

运行结果，如图 8-13 所示.

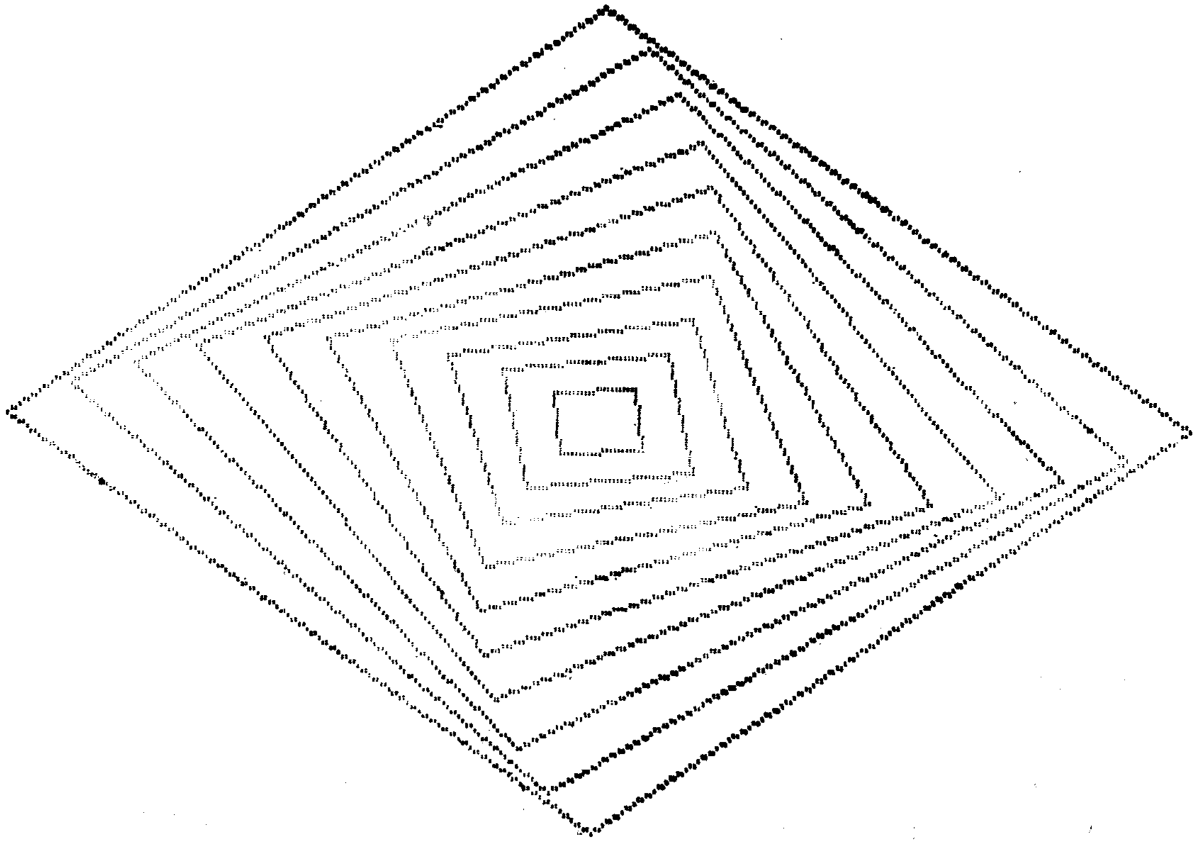


图 8-13

例 2　对矩形作放大变换.

解　可编程序如下:

```
PROGRAM ex08_18
! Make a "square" picture, and draw it.
!
```

```
4  SET   window  -12, 12, -10, 10
5  FOR  k=1  to  10
6        DRAW  square  with  scale(k)
7  NEXT  k
8  END
9  PICTURE  square
10       PLOT  -1, 1;
11       PLOT   1,   1;
12       PLOT   1, -1;
13       PLOT-1, -1;
14       PLOT-1,   1
15  END  PICTURE
```

运行结果，如图 8-14 所示.

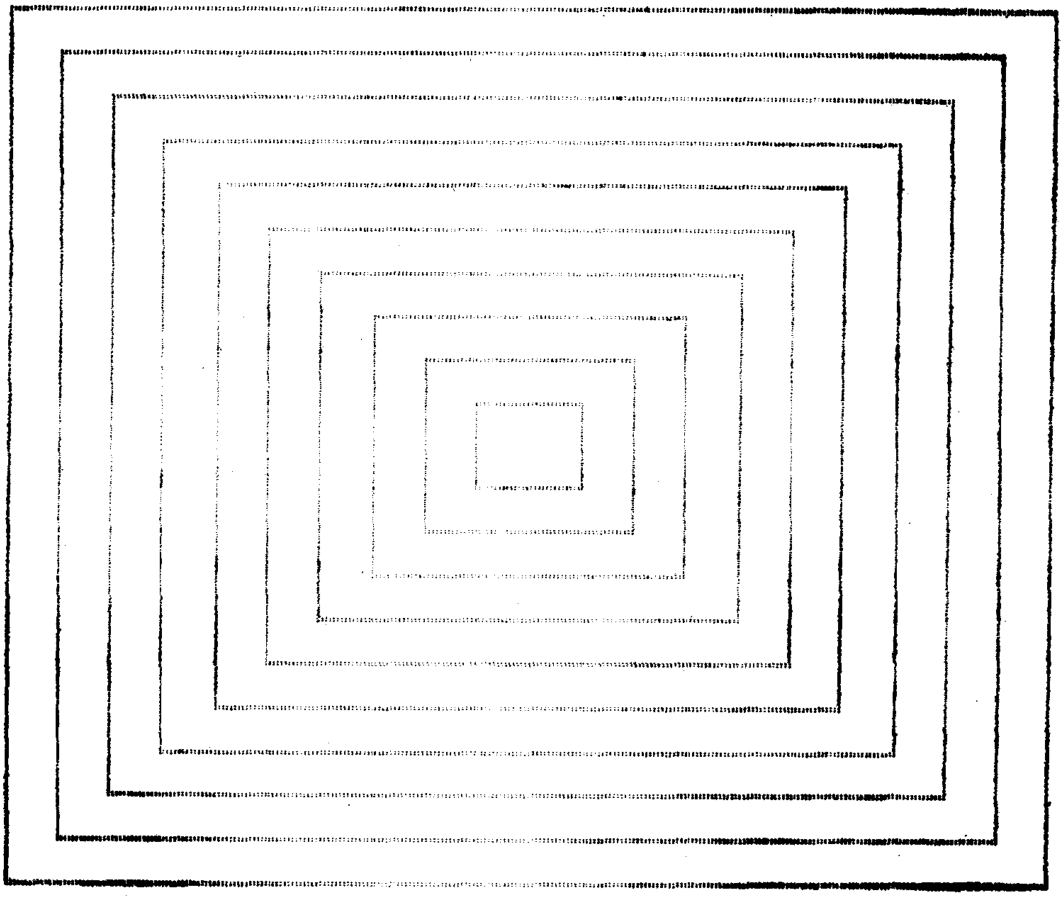


图 8-14

8.7.4 图画调用形式之三

DRAW pic(rp1,rp2,…)with trans*trans…

其中 trans 可以是 True BASIC 内部变换的任一个.

执行它，连续几个变换是自左至右进行，在屏幕上显示的是连续变换的最终结果.

例 1　画出一个房子.

解　画一个房子包括:屋子、屋顶、烟囱、门和窗户. 而且进行了多种的变换，主程序如下:

```
PROGRAM ex08_19
! Draws three houses using pictures
!
SET window 0, 10, 0, 10
SET BACK "green"
OPTION ANGLE degrees
DRAW House with Scale(2)*Shift(7, 6)
DRAW House with Scale(1)*Shift(1.8, 4)
DRAW House with Rotate(25)*Scale(1, 0.6)*Shift(6.2, 1.8)
END
```

House 是一个外部图画，它又调用了另一个 window 图画，而且在调用中也进行了变换，外部图画 House 如下:

```
PICTURE House
    SET COLOR "cyan"
    PLOT AREA: -1, -1; -1, 1; 1, 1; 1, -1  ! The house itself
    !
    SET COLOR "magenta"
    PLOT AREA: -1, 1; 0, 1.5; 1, 1              ! Its roof
    PLOT AREA: -1, 1; -1, 1.4; -.8, 1.4; -.8, 1 ! and chimney

    SET COLOR "white"
    PLOT AREA: -.15, -1; -.15, -.3; .15, -.3; .15, -1   ! A door

    DRAW Window with Shift (-.5, -.5)           ! and 4 windows
    DRAW Window with Shift (-.5, .5)
    DRAW Window with Shift (.5, .5)
    DRAW Window with Shift(.5, -.5)
END PICTURE

PICTURE Window
    SET COLOR "background"
    PLOT AREA: -.1, -.2; -.1, .2; .1, .2; .1, -.2 ! The window
    SET COLOR "white"
    PLOT -.1, 0; .1, 0                          ! and its sill
    PLOT 0, -.2; 0, .2
END PICTURE
```

运行结果，如图 8-15 所示.

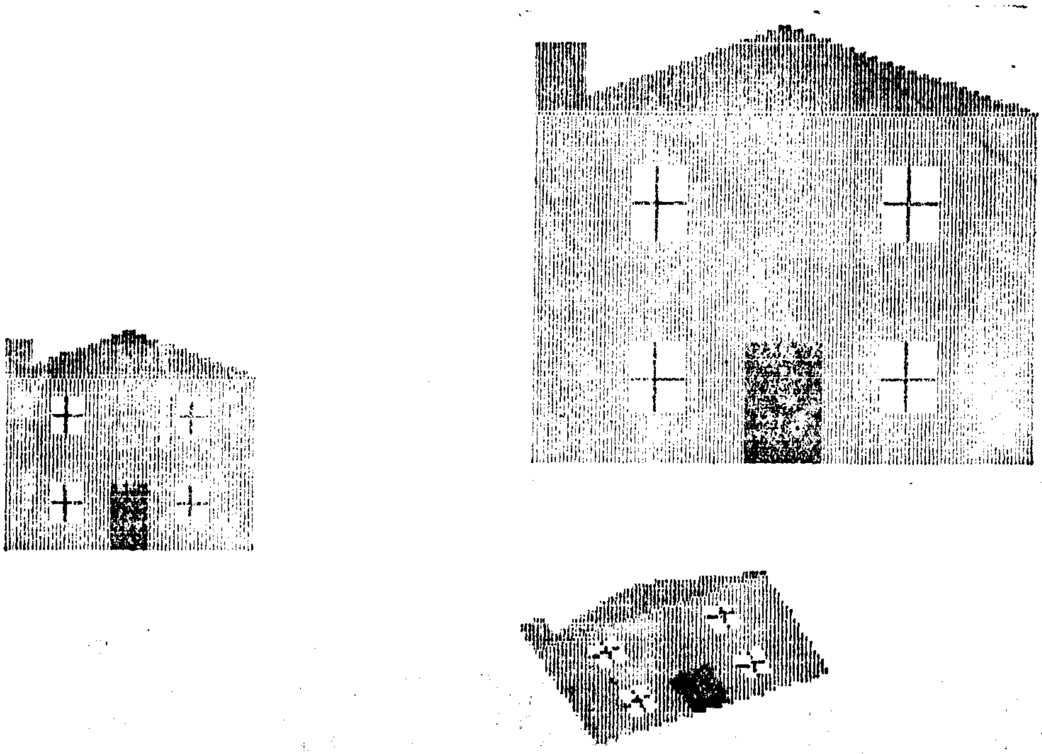


图 8-15

例 2 画一面五星红旗.

解 程序 ex07_3 曾画过一个五角星，再给它充满黄色，可编成一个图画辅程序，然后在主程序中调用并变换它，即可画出一面五星红旗(由于数据资料不足，这仅是一个模拟程序)，具体程序如下:

```
PROGRAM ex08_20

SET window −2.5, 10.5, −7, 3
SET back 17
SET color "red"
BOX AREA −2.5, 10.5, −7, 3
SET color "yellow"
OPTION ANGLE degrees
DRAW star with scale(1, 1.2)
FOR m=0 to 3
      LET fi=−45+m*30
      DRAW star with rotate(fi)*
                shift(1.5*cos(fi), 1.5*sin(fi))*scale(.4, .45)
NEXT m
END
PICTURE  star
      LET bgn=pi/2
```

```
18         LET pio=4*pi/5
19         FOR k=0 to 4
20              LET x=0.8*cos(bgn+k*pio)
21              LET y=0.8*sin(bgn+k*pio)
22              PLOT x, y;
23         NEXT k
24         PLOT 0.8*cos(bgn), 0.8*sin(bgn)
25         FLOOD    0, 0
26         FOR k=0 to 4
27              FLOOD 0.4*cos(bgn+k*pio), 0.4*sin(bgn+k*pio)
28         NEXT k
29    END PICTURE
```

运行结果，如图 8-16 所示.

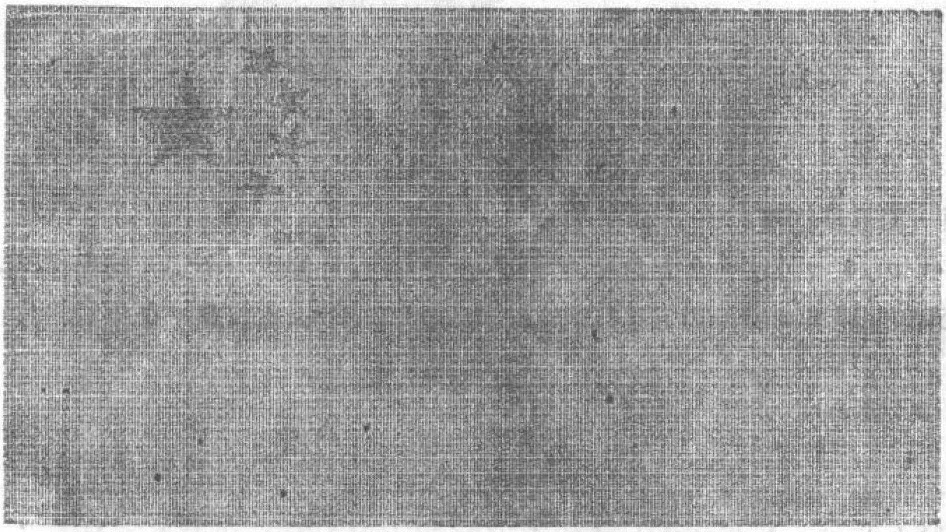


图 8-16

ex08_20 的第 12 行、第 13 行是一条语句，本应写在一行上，这里为了便于印刷才分为两行的.

附注：内部变换实际上是一个 4×4 矩阵右乘一个齐次坐标向量的函数. 如，shift(a, b)即

$$(x,y,0,1)\times\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\a&b&0&1\end{bmatrix}\text{值为}(x+a,y+b,0,1)$$

内部变换与对应矩阵如下：

shift(a,b)	1	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1
scale(a,b)	a	0	0	0
	0	b	0	0
	0	0	1	0
	0	0	0	1
scale(a)	a	0	0	0
	0	a	0	0
	0	0	1	0
	0	0	0	1
rotate(α)	cos(α)	sin(α)	0	0
	sin(α)	cos(α)	0	0
	0	0	1	0
	0	0	0	1
shear(α)	1	0	0	0
	tan(α)	1	0	0
	0	0	1	0
	0	0	0	1

True BASIC 规定,如果用户愿意,可以自己选择一个 4×4 数组,用函数辅程序自定义一个函数,用它来代替内部变换.

8.8 打开、关闭和转移窗口

至此各种图形都是输出在一个覆盖了整个屏幕的窗口里，这个窗口为缺省窗口. 这相当于在程序运行开始,由系统自动执行了如下打开窗口语句:

OPEN ＃0: screen 0,1,0,1

执行它,把通道(＃0)与窗口(0,1,0,1)接通. 其实，True BASIC 允许程序员自由地打开、关闭和转移窗口,可把整个屏幕分为多个窗口显示,使用起来很方便.

8.8.1 有关窗口语句

(1) 打开窗口语句一般形式:

OPEN ＃expr: screen left,right,bottom,top

其中 expr 为数值表达式，其值为通道号,在[1:1000]之间.

left(左),right(右)，bottom(底)，top(顶)是数值表达式,其值是在[0,1]区间上的任一实数. 它们表明窗口放在屏幕上的位置和范围,如图 8-17 所示.

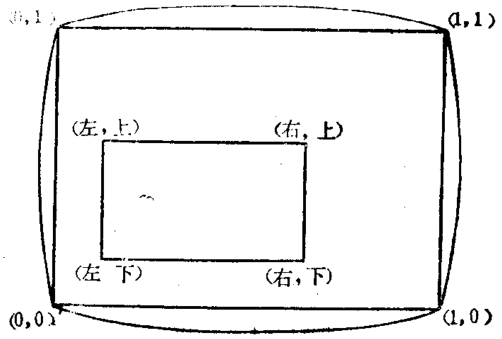


图 8-17

执行 OPEN 语句，即可把一个通道与一个窗口连系起来．程序通过通道可输出到窗口上，通道与窗口是一一对应的，这个对应关系是由 OPEN 语句确定的．在程序运行中只有一个窗口是处于可输出状态，称为当前窗口．当一个新的窗口被打开后，则自动成为当前窗口．为确定(查找)当前窗口的位置可用如下语句询问：

ASK screen Vleft, Vright, Vbottom, Vtop

其中 Vleft, Vright, Vbottom, Vtop 为数值变量.

执行它，系统把当前窗口的坐标位置分别赋给 4 个变量.

要特别注意，新的窗口是否与原有窗口重叠．有些计算机提供的是“透明窗口”，能透过一个窗口看到在它下面的窗口；有些计算机提供的是“非透明窗口”，它盖住了它下面的其它窗口．如果想在几个窗口交替输出，则不要设置重叠窗口.

(2) 关闭窗口语句一般形式：

CLOSE #expr

执行它，则关闭通道．如上所述，True BASIC 的窗口是通过通道进入的，所以，关闭通道即关闭了对应窗口．当关闭一个窗口后，True BASIC 自动转回到缺省窗口．当程序运行结束时，所有打开的窗口会自动关闭.

(3) 窗口转移语句的一般形式：

WINDOW #expr

执行它，expr 对应的窗口变为当前窗口．窗口转移包括原来的颜色、光标位置等的恢复，这些是由系统自动完成的.

注意，expr 通道必须是打开而又未关闭的通道，否则出现错误.

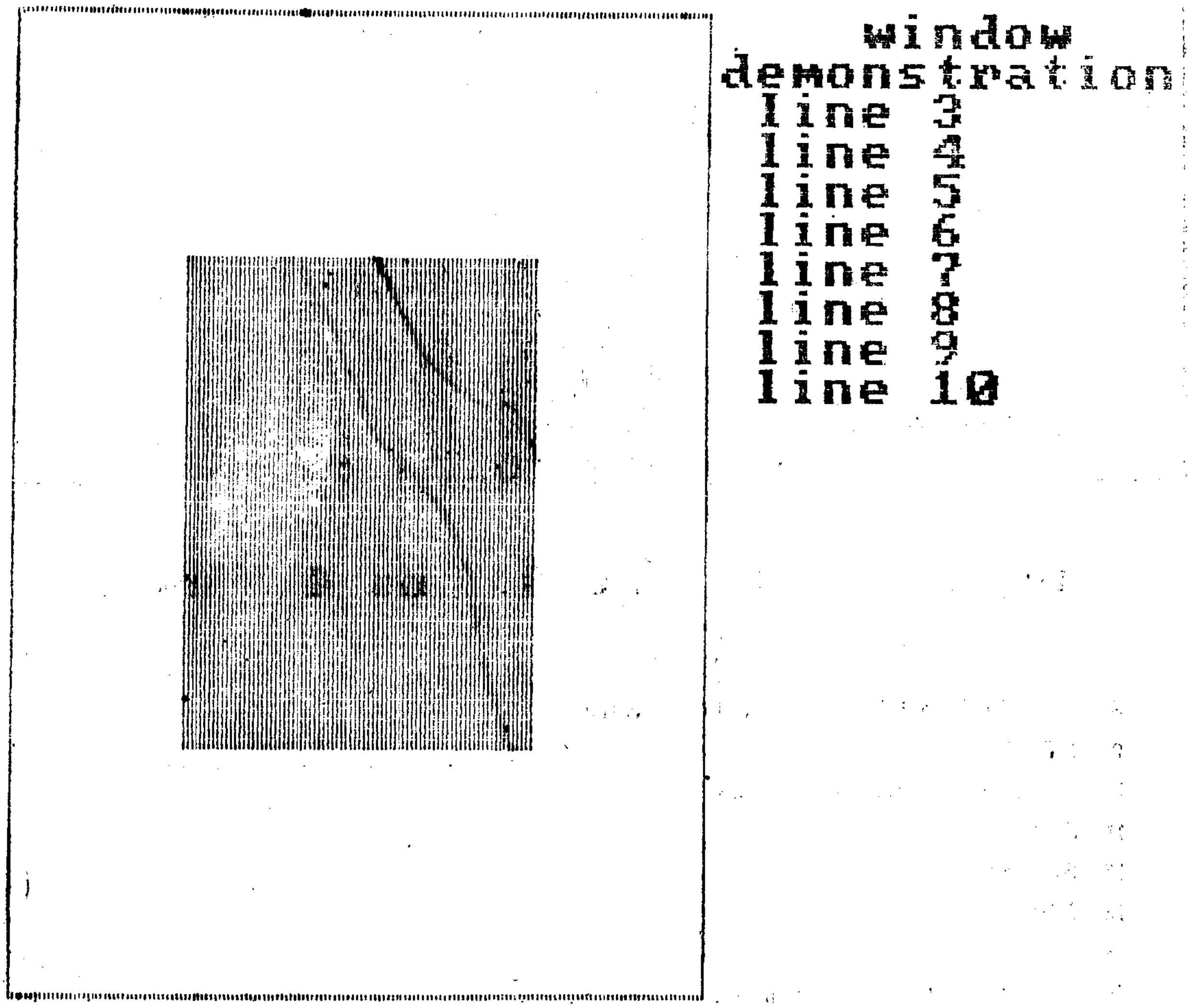
8.8.2 多窗口应用举例

例 阅读如下程序：

```
PROGRAM ex08_21
! window example
!
OPEN #1: screen 0, 0.49, 0, 1
SET window -10, 10, -10, 10
BOX LINES -10, 10, -10, 10
BOX AREA -5, 5, -5, 5
OPEN #2:screen 0.5, 1, 0, 1
SET cursor 1, 5
PRINT "window"
PRINT "demonstration"
FOR i=3 to 10
      SET cursor i, 2
      PRINT "Line";i
NEXT i
```

```
END
```

这个程序打开了两个窗口：一个在左，一个在右. 左边输出一个图，右边输出一些字符，如图 8-18 所示.



例 2 打印直方图，如图 8-19 所示.

解 这个图可分为上下两个窗口，上面的直方图和 ex07_7 程序类似. 为了使用窗口转移语句，把图形与文字分开输出. 按照上述分析可编程序如下：

```
PROGRAM ex08_22
!
OPEN #3 :screen 0, 1, 0.3, 1
SET window 0, 6, 0, 150
BOX LINES 0, 6, 0, 150
FOR x=1 to 5
      READ value
```

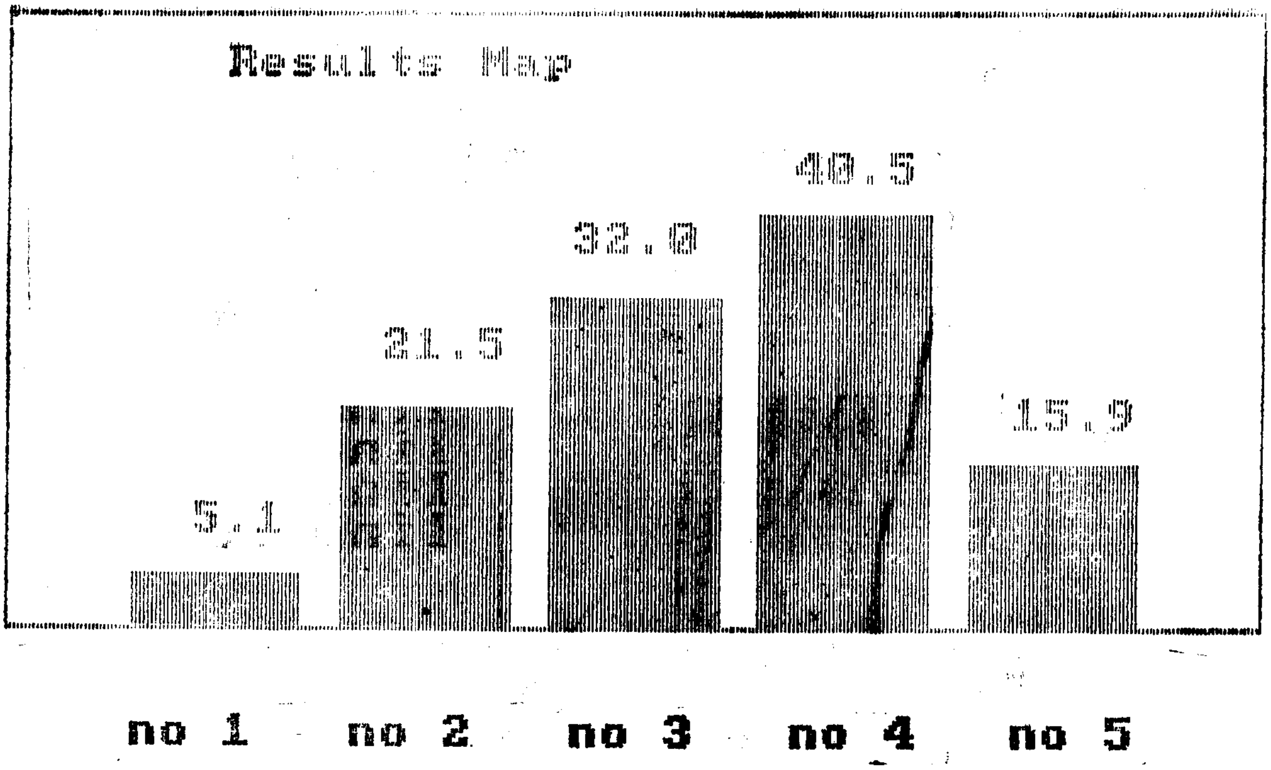


图 8-19

```
8        BOX AREA x-0.4, x+0.4, 0, value
9   NEXT x
10  OPEN #4 :screen 0, 1, 0, 0.29
11  SET window 0, 150, 0, 3
12  SET cursor 3, 2
13  FOR i=1 to 5
14       PRINT "     ";
15       PRINT using "no #":i;
16  NEXT i
17  RESTORE
18  WINDOW #3
19  FOR x=1 to 5
20       READ value
21       PLOT TEXT ,at x-0.4, value+10 :using$("###.#", value)
22  NEXT x
23  !
24  DATA 5.1,21.5,32.0,40.5,15,9
25  END
```

8.9 库 文 件

如前所述,子程序或函数是个独立的程序块,可以单独编译. 具有通用性,可为大家共享. 如,梯形法积分子程序. 为了便于调用,True BASIC 引入了库. 库是一个外部子程序、函数和图画的集合,它没有主程序,它不能直接运行. 但是它的子程序、函数或图画可供其它程序调用. 所以,一个库文件,就象一个"工具箱",其中装着若干个有用的函数、子程序和图画.

8.9.1 库的形式及辅程序的调用

(1) 库的形式:

```
EXTERNAL
   DEF func1         ⎫
     ⋮               ⎪
   END DEF           ⎪
    …                ⎬ n个函数(n≥0)
   DEF funcn         ⎪
     ⋮               ⎪
   END DEF           ⎭
   SUB sub1          ⎫
     ⋮               ⎪
   END SUB           ⎪
    …                ⎬ m个子程序(m≥0)
   SUB subm          ⎪
     ⋮               ⎪
   END SUB           ⎭
   PICTURE pic1      ⎫
     ⋮               ⎪
   END PICTURE       ⎪
    …                ⎬ l个图画(l≥0)
   PICTURE picl      ⎪
     ⋮               ⎪
   END PICTURE       ⎭
```

库存储在磁盘(软盘)上,有一个文件名(即库名). 它的存取、编辑和用户程序文件一样.

(2) 在程序中调用.

只要用如下语句说明需要调用的子程序(或函数,或图画) 所在的库,其它和调用写在程序 END 之后的外部子程序(或函数,或图画)一样.

LIBRARY "filename1","filename2",…

其中 filename 是将要用到的子程序(或函数,或图画)所在的库名. 用到几个库就写几个库名. 注意: 库名两端要加上双引号.

例 有一个名叫 ex08_lib 的库.

```
EXTERNAL
!
```

```
! CHANGE run of spaces into tab,and vice versa.
!
DEF squish$(Line$)
    DO
        LET i=pos(Line$,"  ")
        IF i=0 then EXIT DO
        LET Line$[i:i]=chr$(9)
    LOOP
    LET squish$=Line$
END DEF
DEF unsquish$(Line$)
    LET t$=chr$(9)
    DO
        LET i=pos(Line$,t$)
        IF i=0 then EXIT DO
        LET Line$[i:i]="  "
    LOOP
    LET unsquish$=Line$
END DEF
```

调用程序:

```
PROGRAM ex08_23
!
LIBRARY "ex08_lib"
DECLARE DEF squish$,unsquish$
LINE INPUT ln$
LET wk$=squish$(ln$)
PRINT unsquish$(wk$)
END
```

8.9.2 标准库

在 True BASIC系统中,提供了标准库文件,其中包含四个数学函数库,一个画图子程序库.

(1) 数学函数库.

数学函数库分:

FNHLIB 双曲函数.

FNTLIB 三角函数(以弧度制)

FNTDLIB 三角函数(以角度制)

FNMLIB 数学函数.

这些库以编译形式存放在磁盘上,下面是文件名及函数名表:

原书缺页

原书缺页

注，在程序中使用库中任意一个子程序，只要把语句

LIBRARY "graphlib"

加在任何调用语句之前，再用 CALL 语句调用即可.

习　题

8.1 为如下函数 f(x) 编写一个函数辅程序:

$$f(x)=\begin{cases}2x^2+3x+4, & (x\leqslant 2)\\ -2x^2+3x-4, & (x>2)\end{cases}$$

8.2 首先为如下函数 $y(x)$ 编写一个函数辅程序:

$$y(x)=\begin{cases}1+\sqrt{1+x^2}, & (x<0)\\ 0, & (x=0)\\ 1-\sqrt{1+x^2}, & (x>0)\end{cases}$$

然后，给定a 和z 计算

$$f=2+y(a+z)$$

$$g=\frac{y(\cos(a))+y(z)}{2}$$

8.3 编写一个计算两个整数(I 和 J)最小公倍数的函数辅程序，并用 6，8 验证之.

[提示] (1) 用辗转相除法求最大公约数.

(2) I * J = 最大公约数 * 最小公倍数.

8.4 试编一个计算

$$C_n^k=\frac{n!}{k!(n-k)!}$$

的函数辅程序，然后计算 $C_6^0, C_6^1, C_6^2, C_6^3, C_6^4, C_6^5, C_6^6$ 的值.

8.5 写一个子程序辅程序 norm(a, n, am, sd)，计算数组a的平均值 am 和标准方差 sd.

$$am=\frac{1}{n}\sum_{I=1}^{n}a(I)$$

$$sd=\sqrt{\frac{1}{n}\sum_{I=1}^{n}(a(I)-am)^2}$$

8.6 写一个子程序辅程序，求某个二维数组中绝对值最大的元素和它的下标.

8.7 写一个子程序辅程序，用绝对值最大的元素除以每个元素，使向量(一维数组)规格化. 写一个主程序，调用该子程序.

8.8 编写一个函数辅程序，用于计算二维数组 a(m,n) 中第 l 行或第 l 列元素之和.

[提示] 函数语句可写成

FUNCTION sumroc(a, m, n, l, roc)

其中，roc 为数值变量，当 roc=1 时，计算第 l 行元素之和，否则计算第 l 列元素之和.

8.9 计算 n 阶方阵的模数:

$$\|A\|=\max_{1\leqslant i\leqslant n}\sum_{j=1}^{n}|a_{ij}|$$

以及相应的行号，并用 A(4, 4) 方阵验证之.

[提示] 模数即各行元素之和的最大值.

8.10 写一个计算方阵的转置矩阵的子程序,并用 A(4,4)验证之.

8.11 编写一个计算两个一元 n 次多项式乘积的子程序.

[提示] 以两个一元二次多项式为例:

$$\begin{array}{r} a_1 + a_2x + a_3x^2 \\ +b_1 + b_2x + b_3x^2 \\ \hline c_1 + c_2x + c_3x^2 + c_4x^3 + c_5x^4 \end{array}$$

其中 $c_1=a_1b_1$ $c_2=a_1b_2+a_2b_1$

$c_3=a_1b_3+a_2b_2+b_1a_3$

$c_4=a_2b_3+a_3b_2$

$c_5=a_3b_3$

从中不难发现: c 的下标等于 a 的下标加上 b 的下标再减去 1.

8.12 编写一个画椭圆的图画辅程序,调用它沿 y 轴自下而上的作平移变换.

8.13 编写一个画直线的图画辅程序,调用它沿着一个圆连续作平移变换.

8.14 编写一个画抛物线的图画辅程序,调用它围绕坐标中心作旋转变换.

8.15 编写一个画正五边形的图画辅程序,沿直线 $y=\frac{1}{2}x$ 作放大平移变换.

8.16 编写一个能画不同高度圆锥的图画辅程序.

8.17 编写一个能画不同直径圆柱的图画辅程序.

8.18 编写一个能画不同半径圆球的图画辅程序.

8.19 编写一个动画程序. 如,海鸥飞翔,跳天鹅湖舞等.

第9章 文　件

前面讲述了常量、变量和数组，这些数据都是放在内部存储器(简称内存)里. 由于内存容量有限，无法处理如仓库、银行的帐目等大规模数据问题，和其它高级语言一样，True BASIC 引入了文件 True BASIC 文件是存储在磁盘上的数据集合. 磁盘是一种外部磁性存储器 (简称外存) 文件是独立于程序存在的实体，它的数据通过文件名可以被程序处理.

9.1 文件概述

9.1.1 文件种类

(1) 文件按其结构分为三种:

正文文件: 它是由一行一行可图示的字符组成.

记录文件: 它是由一个一个记录(数据项)组成的 在一个记录文件中，各个记录长度相同. 各记录有记录号，可随机存取

字节文件: 它是由一个一个字节组成，一个字节有 8 位，可存放一个字符.

(2) 文件按存取方式分为二种:

顺序文件: 正文文件和字节文件属于顺序文件，只能依顺序存取.

随机文件: 记录文件属于随机文件，可按记录号设定文件指针，进行存取.

除此之外，屏幕、窗口、打印机也作为文件处理.

9.1.2 通道与文件

程序与文件没有固定的联系. 一个程序在处理某个文件之前，要用"通道"把程序与文件连接起来. 通道是用一个号码标识符(#)后跟一个数值表达式表示，如，#3. True BASIC 对表达式作四舍五入取整，其值要在 1～999 之间，否则会出现错误信息. True BASIC 规定: 通道 #0 表示屏幕和键盘，此通道总是打开的，不能 通往文件的通道一经打开，直到关闭之前，该通道就代表它对应的文件.

在一个 True BASIC 程序中，最多可同时打开 10 个文件. 注意，各机器操作系统允许打开的文件多少不尽相同，请查阅有关用户手册.

9.1.3 文件指针的位置

如前所述，文件是数据的集合. 程序与文件之间每次只传送一个数据(一行，一个记录或一个字节). 为确定传送数据位置引入了文件指针，用它标志文件当前位置. 也就是即将输入/输出数据的位置.

文件指针随着文件数据的输入/输出自动地沿着从文件开始(begin)到结束(end)方向逐步移动．当从文件中取出数据时，文件结束位置是不变的，而存入数据时，文件结束位置则是变化的。

文件指针可以用如下语句重新设置(移动)：

SET #expr: pointer position

或

RESET #expr: position

其中 expr 为数字表达式，值四舍五入取整为通道号．SET，RESET 和 pointer 为关键字．position 为指针位置，各种文件允许取的值如表 9-1 所示．

表 9-1

正文文件	记录文件	字节文件	含　义
begin	begin	begin	第一个记录
end	end	end	最后一个记录的后面
	next		下一个记录
	same		前一个记录

此外，对记录文件还可用如下语句：

RESET #expr:record rn

或

SET #expr:record rn

其中 rn 为数值表达式，值四舍五入取整为记录号．record 为关键字．

执行它，把文件指针移到第 rn 号记录．

文件指针当前位置，可以用如下语句询问：

ASK #expr:pointer var$

或

ASK #expr:record var

其中 var $ 是字符串变量，它的值是"begin"，"middle"或"end"．

var 是数值变量．对正文文件和字节文件，其值表示当前指针离开起点的字节个数，对于记录文件，其值表示当前指针离开起点的记录个数．

9.1.4 文件数据的测试

在读取一个文件的数据时，要经常测试文件指针是否已达到末尾．为此 True BASIC 引入了文件数据测试条件如下，这和 DATA 数据测试类似．

more #<expr>　文件还有数据时为真，否则为假．

end #<expr>　文件指针已到末尾(即已没有数据)为真，否则为假．

9.1.5 文件存取模式 (access)

可以在 OPEN(打开)语句中，用 access<accmode$>来设置文件存取模式．有如下三种形式：

input　　只读

output　只写

outin　　可读可写（缺省形式）

如果在 OPEN 语句中，access 项缺省，即 outin 形式. 对一个打开的文件，可以用如下语句询问它的存取模式:

ASK ＃expr:access var$

执行它，文件存取模式，"input"，"output"或"outin"可赋值给字符串变量 var$.

9.1.6　文件建立模式(create)

可以在 OPEN 语句中，用 create <crtmode$>来设置. 有如下三种模式:

new　建立新的文件

old　用已存在的文件 <缺省形式>

newold　用已存在的文件，或建立新文件.

这是不能询问的，也不需要询问.

9.1.7　文件的组织模式(organization)

可在 open 语句中，用 organization <orgmode$>来设置. 有如下三种模式:

text　　正文文件

record　记录文件

byte　　字节文件

这部分不能缺省，可用如下语句询问打开文件的组织形式:

ASK ＃expr:organization var$.

执行它，文件组织模式，"text"，"record"，"byte"或"window"，可赋值给字符串变量var$.

9.1.8　文件的大小

文件的大小随输入数据的过程而变化，可以用如下语句询问

ASK ＃expr:filesize var

其中 var 是数值变量. 它的值对于正文文件和字节文件表示字节个数，对于记录文件表示记录个数.

9.2　文件的通用语句

如上所述，文件有三种，对于这些文件，有如下四种语句是通用的:

OPEN　　打开某磁盘文件通道.

CLOSE　关闭某通道.

ERASE　删除文件内容，使成为一个空文件.

UNSAVE 删除某个文件.

9.2.1 OPEN 语句

一般形式:

```
OPEN #expr:NAME expr$,ACCESS  accmode$,
                      CREATE  crtmode$,
                      ORGANIZATION orgmde$,
                      RECSIZE recsize
```

其中 expr 为数值表达式,其值为通道号,在[1:999]之间.

expr$ 为字符表达式,其值为文件名,以字母打头,限定名字长不超过 8 个字符.

[,ACCESS accmode$] 为可选项,如 9.1.5 所述.

[,CREATE crtmode$] 为可选项,如 9.1.6 所述.

ORGANIZATION orgmode$ 为必选项,如 9.1.7 所述.

[,RECSIZE recsize] 为可选项,对记录文件可选,对正文文件不可选, 对于字节文件的读态可选.

执行此语句,打开与磁盘文件 expr$ 相关联的通道 expr, 并使文件指针指向文件开头(Begin). 例如,下面例子均为合法的:

```
OPEN #1:name"file1",access input,organization text
OPEN #i:name"file2",access"input",organization"text"
OPEN #k:name"file3",access acc$,organization org$
```

这里 i,k,acc$,org$ 是"变量引用",即对被赋过值变量的使用. 表示文件属性的字符串,可以不带引号. 这三个文件都是已存在的文件.

9.2.2 CLOSE 语句

一般形式:

```
CLOSE #expr
```

其中 #expr 是文件的通道号. 执行它,关闭 #expr 通道, 也就是将通道 #expr 与相关联的文件断开. 在执行 CLOSE 语句之前,通道可以是打开的, 也可以是关闭的. 执行 CLOSE 语句之后,通道可以重新和另一个文件建立关联.

9.2.3 ERASE 语句

一般形式:

```
ERASE #expr
```

执行它, 删除与 #expr 关联文件的内容,即成为空文件,文件的开头和结尾重合, 文件指针可以说指向开头. 注意: 它的文件组织模式也随之不存在了. True BASIC 规定在下次使用此文件时,要重新给它定义组织模式.

9.2.4 UNSAVE 语句

一般形式:

UNSAVE expr$

执行它，删除文件名为 expr$ 的文件. 如果文件存在，就被删除，否则 True BASIC 认为出错.
注意: 不能用 UNSAVE 语句删除已打开的文件，因为这时可能引起意想不到的后果.

9.3 正 文 文 件

正文文件是由一行一行的可图示的字符组成. 从多方面来看，这种文件实质上是屏幕的映象，和屏幕显示有许多类似的地方.

9.3.1 正文文件的范围与域宽

正文文件的范围可用如下语句设置(修改)和询问:

SET #expr MARGIN expr(范围)

ASK #exp MARGIN var

SET #expr ZONEWIDTH expr(域宽)

ASK #expr ZONEWIDTH var

其中 expr 为数值表达式，四舍五入取整.

正文文件的 MARGIN 和 ZONEWIDTH 可在任何时候设置，并立即获得结果.

正文文件和屏幕一样，其范围和域宽的内定值分别为 80 和 16.

9.3.2 正文文件的读写

一个正文文件是一个顺序文件，在作存取操作时，要切实注意指针位置.

(1) PRINT 语句的一般形式:

PRINT #expr (存入一个空行)

PRINT #expr: expr1, expr2, …

PRINT #expr, USING format$:items

MAT PRINT #expr, USING format$:items

执行上述语句把表达式 expr 或项 items 的值存入与通道#expr 相关联的文件之中. 注意: 由于正文文件不能重写数据，所以在执行 PRINT 语句之前，必须用设置指针语句使文件指针指向文件末尾，以便将新写入的数据放在原有数据之后.

例如，把一个帐单(bill)输入正文文件.

```
PROGRAM ex09_1
! open text file, and print bill.
!
OPEN #1:name "bill", organization text, create new
PRINT #1
```

```
PRINT #1:'Bill for November'
PRINT #1
LET format $="<################  $##,###.##'
!
PRINT #1,using format$:'li ping', 23.50
PRINT #1,using format$:'lu jian',55.70
PRINT #1,using format$:'wang gang',17.00
PRINT #1,using format$:'gao feng',1200.00
PRINT #1
PRINT #1,using format$:'total=',23.5+55.7+17+1200
PRINT #1
END
```

(2) INPUT 语句的一般形式:

INPUT #expr:var1,var2,…

LINE INPUT #expr:var1$,var2$,…

MAT INPUT #expr:array1,array2,…

MAT LINE INPUT #expr:array1$,array2$,…

执行上述语句,从与通道 expr 相关联的正文文件中取出值,依次赋给相应的变量或数组.

True BASIC 从被打开的正文文件中取数据,和从键盘上键入一样,不同的是:从键盘上输入如果出错,可以重新输入;而从文件输入出错会导致致命的后果.为避免错误出现,切实注意存取的一致性,即如何存入,也如何取出.

正文文件可以打印和显示.此外 True BASIC 把打印机看作是一个特定的正文文件,命名为:prn.存取方式为 output.如

OPEN #1:name"prn",access output

9.3.3 正文文件应用举例

试编一个程序,输入任一单词到一个正文文件中去查找,然后作出回答,若没有找到,则把新单词存入文件末尾.

```
PROGRAM ex09_2
! ask for words from the user, and lookup them up in a dictionary.
!        If not found, let the user add them to the end of the file.
!
DECLARE DEF lookup
OPEN #1:name'dictionary', access outin, organization text
DO
    PRINT'word'
    LINE INPUT word$
    IF word$ <> '' then
        IF lookup (word$)=0 then
```

```
            PRINT"Not found."
            CALL add(word$)
        ELSE
            PRINT "Found."
        END IF
    END IF
LOOP until word$=" "
STOP
!
! Lookup__ See if word is in dictionary.
!     return 1 if found, 0 otherwise.
!
DEF lookup(w$)
      SET #1:pointer BEGIN
      LET w$=lcase$(w$)
      DO while more #1
          INPUT #1:d$
          IF w$=d$ then
              LET lookup=1
              EXIT DO
          END IF
      LOOP
END DEF
! Add__ add word to end of dictionary,
!     if user thinks it should be added.
SUB add(w$)
      PRINT "Should it be added to the dictionary";
      INPUT yesno$
      IF lcase$(yesno$[1:1]) < > "y" then EXIT SUB
      SET #1:pointer END
      PRINT#1: lcase$(w$)
END SUB
END
```

这个程序包含两个内部子程序，一个 lookup(查找)子程序；另一个 add(添加)子程序，如果用户想存入(回答 yes)，可以把在文件中没有找到的单词添入。

9.4 记录文件

True BASIC 记录文件与其它高级语言(如 COBOL，PASCAL)的记录文件结构不同，True

BASIC 记录文件虽然是由一个一个“记录”组成的，每个记录有一个记录号，但每个记录只有一个数据项，且数据项的类型可以不同．这样，在同一个记录文件中，各个记录的数据项类型可以是不一样的．记录文件与指针如下所示：

记录号	文件	指针位置
1	"Li Ping"	begin
2	21	
3	"male"	
4	"201.5599"	←same
5	"Lu Jian"	next
6	26	
7	"female"	
8	"87.5831"	
		end

设置记录文件指针语句一般形式：

SET #expr:POINTER recorde

其中 recorde 是数值表达式，四舍五入取整，为记录号．

执行它，使文件指针指向第 recorde 记录．

记录文件的各个记录都是以内部存储(二进制)形式存储的，所以，记录文件不能直接在屏幕上显示．

9.4.1 记录长度的设置与询问

设置记录长度语句一般形式：

SET #expr:RECSIZE exprl

其中 expr 为通道号，exprl 为记录长度表达式．

执行它，可以设置(或修改)记录文件的记录长度．

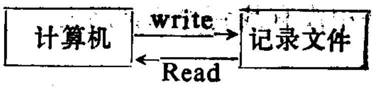
询问记录长度语句一般形式：

ASK #expr RECSIZE var

其中 var 为数字变量．

执行它，可以询问记录文件的记录长度．对于正文文件和字节文件其值为 0.

9.4.2 记录文件的读写



(1) WRITE 语句一般形式．

WRITE #expr:exprl,expr2,…

MAT WRITE #expr:arrayl,array2,…

执行它，可以把字符串或数值数据混合写入与通道 expr 相关联的文件中．

True BASIC 依照当前指针位置顺序地写入各个记录，每写入一次(项)，文件指针向前移动一次，记录编号加 1. 数组各元素按计程器顺序依次接连写入. 通过移动文件指针，对记录文件从指针当前位置开始向后，进行覆盖重写(修改)，或在记录文件末尾添加，以使文件扩充.

如果所使用的相应通道没有打开，或者打开的是只读文件，这时使用 WRITE 语句就会引起出错.

例如，写某一门课程的成绩表到一个记录文件中. 文件名为 result.

```
PROGRAM ex09_3
!    Write name and number to record file.
!
INPUT prompt "file$?": file$
OPEN #1: name file$, organization record    ! , create new
!    SET #1 :recsize 20
DO while more data
    READ name$, number
    WRITE #1:name$, number        (一次写入两个记录)
LOOP
DATA Li Ping,90, Lu Jian, 70, Wang Gang,50
DATA Gao Feng, 100
END
```

(2) READ 语句的一般形式:

READ #expr:var1,var2,…

MAT READ #expr:array1,array2,…

执行它，从文件指针当前位置的记录取值赋给 var1，然后文件指针自动移到下一个记录，再取值赋给 var2，如此继续，直至本语句变量均被赋值. 数组各元素按计程器顺序一一被赋值. 如果数据不够，即文件指针已移到文件末尾，还要取数据，就会出现读错. 另外，通道没有打开，或打开的是只写文件也会出错.

例如，打印 ex09_3 生成的文件 result.

```
PROGRAM ex09_4
! Read file written by previous program.
!
OPEN #1:name "result", organization record
DO while more #1
    READ #1:name$, number     (一次读出两个记录)
    PRINT name$,
    PRINT number
LOOP
END
```

运行结果

```
Li Ping        90
Lu Jian        70
Wang Gang      50
Gao Feng       100
```

9.4.3 记录文件的应用举例

(1) 多数据项记录的合并与分解.

如前所述，True BASIC 记录文件的每一个记录只有一个数据项. 如果建立一个雇员状况的数据文件，每个雇员有：姓名、职称、年龄、出生日期和工资 5 项数据，又如何处理呢？

如果把 5 项分别放在不同的记录之中，会引起两个问题：(一) 查找某个数据项比较困难. 例如第一个雇员的工资是第 5 号记录，第二个是第 10 号记录，等等. (二) 因为各记录长度相同，这就造成存储空间很大的浪费.

解决的办法是将几个数据项放在一个记录中. 具体地说，就是把一个雇员的所有属性，构成一个字符串，放在一个记录中. 让每个属性在记录中都有固定的位置，以便于数据存取和删除. 这要建立两个辅程序. 一个是 pack$ 函数，把所有属性压缩成一个字符串. 再建立一个子程序 uppack，把一个字符串分解为各个属性. 具体程序如下.

```
PROGRAM ex09_5a
! Demonstrate packed reoords.
!
DEF pack$(name$, title$, age, start, salary)
     LET pack1$=name$ & title$ & num$(age)
     LET pack$=pack1$ & num$(start) & num$(salary)
END DEF
SUB unpack (s$, name$, title$, age, start, salary)
     LET name$=s$[1:20]
     LET title$=s$[21:40]
     LET age=num(s$[41:48])
     LET start=num(s$[49:56])
     LET salary=num(s$[57:64])
END SUB
!
OPEN #1:name'employ', organization record ! , create new
ERASE #1
SET #1:recsize 86    !Each record is 64 chars long
DO while more data
    READ name$, title$, age, start, sal
    WRITE #1: pack$(name$, title$, age, start, sal)
LOOP
RESET #1:begin
DO while more #1
```

```
      READ #1:s$
      CALL unpack(s$, name$, title$, age, start, sal)
      PRINT name$; title$; sal
LOOP
DATA "Li     ping              ", teacher              ",45,1942,87
DATA "Lu     Jian              "," Epert               ",48,1939,113
DATA "Wang Gang                "," Worker              ",35,1952,200
DATA "Gao Feng                 "," Consul tant         ",75,1912,300
END
```

运行结果:

```
Li     ping              teacher       87
Lu     Jian              Epert         113
Wang Gang                Worker        200
Gao    Feng              Consultant    300
```

在上述程序的 DATA 语句中,字符串长度是固定的. 事实上,输入的字符串长度往往是不等的,还要人为的补上若干空格十分不便,这个问题可采用如下函数和语句解决:

```
DEF sn$(sf$,n)=(sf$&repeat$(" ",20))[1:20]
LEF name$=sn$(name$,20)
LEF title$=sn$(title$,20)
```

这样可重编程序如下:

```
PROGRAM ex09_5b
! Demonstrate packed records.
!
DEF sn$(sf$,n)=(sf$ & repeat$(" ",20))[1:20]
DEF pack$(name$, title$, age, start, salary)
      LET name$=sn$(name$, 20)
      LET title$=sn$(title$, 20)
      LET pack1$=name$ & title$ & num$(age)
      LET pack$=pack1$ & num$(start) & num$(salary)
END DEF
SUB unpack(s$, name$, title$, age, atart, salary)
      LET name$=s$[1:20]
      LET title$=s$[21:40]
      LET age=num(s$[41:48])
      LET start=num(s$[49:56])
      LET salary=num(s$[57:64])
END SUB
!
OPEN  #1:name "employ", organization record !, create new
ERASE #1
```

```
SET    #1:recsize 86                    !Each record is 64 chars long
DO while more data
    READ name$, title$, age, start, sal
    WRITE #1:pack$(name$, title$, age, start, sal)
LOOP
RESET #1:begin
DO while more #1
    READ #1:s$
    CALL unpack s$,name$, title$, age, start, sal)
    PRINT name$; title$; sal
LOOP
DATA Li    ping, teacher, 45,1942,87
DATA Lu    Jian, Epert,    48,1939,113
DATA Wang Gang,Worker, 35,1952,200
DATA Gao Feng, Consultant, 75,1912,300
END
```

(2) 记录文件直接插入分类法.

记录文件是随机文件，可用设置指针语句使文件指针指向任一记录，然后对记录进行存取. 显然，要先存后取，否则读取会出现“非数据项”错. 下面我们利用记录文件可随机存取的特点，写一个“编纂字典”的程序. 它将包括：打印、插入、删除和退出的功能. 具体程序如下：

```
PROGRAM ex09_6
!
INPUT prompt 'file$ ': file$
OPEN #1:name file$,organization record, create new
SET #1:recsize 10
DO
   PRINT '        1:print, 2:insert, 3:delete, 4:end'
    INPUT k
    SELECTCASE k
    CASE 1
       CALL printer(#1)
    CASE 2
       CALL count(#1,n)
       CALL insert(#1,n)
    CASE 3
       PRINT '    31:ask, 32:direct'
       PRINT '        Plase reader answer.'
    CASE 4
       STOP
    CASE else
```

```
    END SELECT
LOOP
END

SUB printer(#99)
   RESET #99 :begin
   DO while more #99
      ASK #99: record VAR
      READ #99:s$
      PRINT var, s$
   LOOP
   PRINT
END  SUB
!
SUB count(#97, recnum)
   SET #97:pointer BEGIN
   LET recnum=0
   DO while more #97
      READ #97 :s$
      LET recnum=recnum+1
   LOOP
   PRINT "old                n=":recnum
END SUB
!
SUB insert(#96, recnum)
  DO
     PRINT "                " "     ":exit."
     INPUT prompt"new$? ":new$
     IF new$ < > " " and new$< >" " then
       RESET #96:record 1
       IF recnum=0 then
         WRITE #96:new$
         LET recnum=recnum+1
       ELSE
         DO
           ASK #96:record PT
           READ #96:old$
           IF new$ = old$ then
             PRINT "record already exists."
             EXIT DO
```

```
          ELSE if new$ < old$ and pt <=recnum then
            CALL movenull(#98,pt,recnum)
            RESET #98:record pt
            WRITE #98:new$
            LET recnum=recnum+1
            EXIT DO
          ELSE if new$ > old$ and pt=recnum then
            RESET #98:record recnum+1
            WRITE #98:new$
            LET recnum=recnum+1
            EXIT DO
          END IF
        LOOP
      END IF
    END IF
    LOOP until new$=' '
  END SUB
  SUB movenull (#98,nul,recnum)
    FOR i=recnum to nul step i
        RESET #98:record i
        READ #98:s$
        RESET #98:record i+1
        WRITE #98:s$
    NEXT i
  END SUB
```

运行过程是会话式的，一次运行过程如下：

```
file$ file2                                  (输入文件名)
    1:print, 2:insert, 3:delete, 4:end
? 2                                          (选择插入功能)
old          n=0
     ' ':exit.
new$? bb
     ' ':exit.
new$? cc
     ' ':exit.
new$? aa
     ' ':exit.
new$? cc
record already exists.
     ' ':exit.
new$? AAA
```

```
    " ":exit.
new$? A
    " ":exit.
new$? " "                                   (输入空格结束插入)
      1:print, 2:insert, 3:delete, 4:end
? 1                                         (选择打印功能)
 1          A
 2          AAA
 3          aa
 4          bb
 5          cc
       1:print, 2:insert, 3:delete, 4:end
? 3                                         (选择删除功能)

    31:ask, 32:direc
      Plase reader answer.
      1:print, 2:insert, 3:delete, 4:end
? 4                                         (选择结束功能)
Ok.
```

这个程序由一个主程序和 4 个辅程序组成. 主程序是一个控制模块, 4 个辅程序分别是打印、统计、插入和移动功能模块. 其中插入模块又有检查是否已存入的功能, 如果没有, 则按字典顺序插到适当位置, 这个模块相当有用. 这个程序只是个模拟程序, 不少地方存在着缺陷, 如, 数字、字符串或包含数字的字符串等一概可以存入. 另外删除模块还留给读者去完成. 当然, 在确定删除记录号之后, 将后边各记录向前移, 编程是容易的.

通过上述二个例子可以看出 记录文件在事务处理、建立知识库等方面将是非常有用的.

9.5 字节文件(Byte File)

字节文件是由一个一个字节组成的, 一个字节放一个字符. 不难想到: 一切程序都可以看作是一个字节文件. 另外, PASCAL 的正文文件, FORTRAN 的 ASCII 码数据文件等 都是 字节文件. 所以说, 字节文件是使 True BASIC 程序获得"原始数据"文件的另一种手段.

字节文件也象其它文件一样, 可以打开、关闭、删除, 也可以设置文件指针指向文件开始或末尾. 值得注意的是: 字节文件可以一个字符一个字符的存取(即字符串长度为 1), 也可以一个字符串一个字符串的存取(即字符串长度大于 1), 而且可以随时改变.

9.5.1 字节文件的读写

字节文件是顺序文件, 每读或写一个字符, 指针自动向下移动一个字节.

(1) 写语句的一般形式:

WRITE #expr:expr1$,expr2$,…

其中 expr 为数值表达式,值为通道号.

expr$ 为字符串表达式. 字节文件只能存入字符串.

执行它,把各字符串表达式的值(字符串)依次存入与通道 expr 相关联的字节文件中去.

(2) 读取语句的一般形式.

字节文件的数据也只能赋给字符串变量,而且,无论何时从字节文件中读取数据, 都必须先确定将要读取多少字符赋给字符串变量. 读语句一般形式如下:

SET # expr:RECSIZE expr1

READ # expr:var1$,var2$,…

或

READ # expr,BYTES expr1:var1$,var2$,…

其中

expr 为数值表达式,值为通道号.

expr1 为数值表达式,值为字节数(即字符串长度).

执行它,按指定字符串长度,从字节文件中读取字符到字符串变量中,直到各变量均被赋值. 若读到文件末尾,但还没完成赋值操作时,则会引起运行出错. 一般地说,读取与写入要一致.

9.5.2 应用举例

例 1 用字节文件输入/输出一张学生成绩表.

解 前面 ex09_3 曾用记录文件描述过. 这里是用字节文件, 请读者注意两个程序的差别, 从而掌握不同文件的用法.

```
PROGRAM ex09_7
! Write name and number to byte file.
!
INPUT prompt "file$?": file$
OPEN #1: name file$,organization byte ! , create new
DO while more data
    READ name$, number$
    LET name$=(name$ & repeat$(" ",20))(1:20)
    LET number$=(number$ & repeat$(" ",5))(1:5)
    WRITE #1:name$,number$
LOOP
SET #1:pointer BEGIN
DO while more #1
    READ #1,bytes 20:name$
    READ #1,bytes 5:number$
    PRINT name$, number$
LOOP
DATA Li Ping, 90, Lu Jian, 70, Wang Gang, 50
```

```
DATA Gao Feng, 100
END
```

运行结果:

```
file$? ff
Li Ping                          90
Lu Jian                          70
Wang Gang                        50
Gao Feng                         100
```

这个结果和用记录文件一样,说明这里字节文件的使用是正确的. 主要差别:字节文件只能存取字符串,记录长度是可变的,而且字节文件是 ASCII 码文件,即可读的.

例 2 试编一个程序,统计一个文件中各元音(A,E,I,O,U)出现的次数.

解 把文件中的字符一个一个的取出,分情况作统计. 可编程序如下:

```
PROGRAM ex09_8
! count vowels in a file
!
INPUT prompt "file: ":file$
OPEN #1:name file$, organizatian byte
SET #1 :recsize 1
DO while more #1
    READ #1: letter$
    SELECT CASE letter$
    CASE "a", "A"
         LET a=a+1
    CASE "e", "E"
         LET e=e+1
    CASE "i", "I"
         LET i=i+1
    CASE "o", "O"
         LET o=o+1
    CASE "u", "U"
         LET u=u+1
    CASE else
    END SELECT
LOOP
PRINT "A="; a,"E=";e, "I="; i, "O=";o, "U="; u
END
```

这里用了设置字符串长度语句,记录长度为 1. 统计时,元音字母的大写与小写不分.

习　题

9.1 编写一个程序(或子程序)分别找出正文文件中长度最长的或最短的行.

9.2 编写一个程序,统计一个正文文件中每个字母出现的次数.

9.3 编写一个子程序,用它检索一个给定值是否在给定文件之中.

9.4 编写一个程序,将一个班的学生的姓名、地址、电话号码写到一个正文文件中去.

9.5 编写一个程序读入一个班学生的姓名、四门功课的成绩到一个文件中,并统计总分在320分以上的学生人数.

9.6 有两个记录文件f1和f2,文件记录的项有学生的姓名和入学总分. f1为男学生,f2为女学生组成的文件,且f1和f2是按入学总分从大到小的有序文件. 编写一个程序,读入这两个文件f1和f2,并将它们合并成第三个文件,仍按入学总分递增顺序排列.

9.7 编写一个程序,可以查找并修改文件中各记录中第n个分量.

9.8 某单位工作人员健康状况简表包括如下几项:

姓名　　　　编号　　　　年龄　　　　性别　　　　健康状况

假定已建立了一个记录文件. 编写一个程序,将文件按性别分为men和women两个文件. 文件记录仅包含姓名、编号、年龄、健康状况. 并要求打印出年龄在40岁到50岁之间的工作人员的记录.

第10章 音　响

为了让计算机能够演奏简单的乐曲或发出其它类型的声响,True BASIC 引入了 PLAY 和 SOUND 两个语句

某些计算机可能没有演奏乐曲或发声的电路,在这些计算机上 True BASIC 则忽略 PLAY 和 SOUND 语句. 因此在编写演奏或发声程序之前，请先检查一下用户指南，看是否有发声功能

10.1 PLAY语句

True BASIC 的 PLAY 语句能使计算机演奏简单的乐曲,它的一般形式是:

PLAY music$

其中 music$ 是一个包含着音乐命令和选择曲调的字符串表达式. 执行它，将按照字符串表达式演奏乐曲.

True BASIC 的音乐命令是:

Tn(速度)设置速度，或乐曲的总的速度. 数字 n 表示在一秒钟里演奏的 4 分音符的个数. n 在 32 到 255 之间. 如果缺省,n=120. 这是一个节拍器的标准速度. 如果要使曲调变快，则增大 n;要使曲调变慢,则减小 n.

On(音阶)设置当前音阶,即用它为后面的音符确定在第 n 个八度上. 音阶是按钢琴键盘来决定的. 为 0 到 7,共有 8 个音阶. 某些机器可能不具有处理全部音阶的能力. 缺省是第 4 个音阶.

>和**<**向上(>)或向下(<)移动一个音阶. 如果在第 4 个音阶上演奏一个乐曲,又执行了">",则下面的音符将用第 5 个音阶演奏. 如果想从 0 音阶向下，或从 7 音阶向上移动，True BASIC 将给出一个出错信息.

A～G(音符) 所有的发音都用 A 到 G 的符号来表示. 其它的代码表示 True BASIC 应如何演奏这些音符的必要信息. 音符之后跟随#或+表示升半个音，跟随－表示降半个音. 单个音符表示用当前音阶和速度演奏一个音符. 在任意一个音符之后可设置音长，用以确定那一个音符的音长,如 A4 表示音符 A 演奏一个 4 分音符. 也可以在一个音符之后打一个点,表示增加它的音长,详见"·".

Ln(音长)设置随后音符的音长. n 在 1 到 64 之间. 音符的音长是 1/n. 即是说，L1 音长是全音符,L2 是一个半音符,L3 是一个三分音符,L4 是一个 4 分音符,以此类推. 调整速度(T),

改变所有音符的实际音长，缺省音长是一个 4 分音符. 在单个音符后面给出音长，只改变那一个音符的音长. 详见 A～G.

(附点)在一个音符后面跟一个附点，表示该音符是按附点音符演奏，它的音长增加 1/2，也就是说，整个音符的音长乘以 3/2，也可以用几个附点来延长一个音符.

Rn 或 **Pn**(休止)休止音长 n. 音长和上述 L 命令中的音长一样量度. 可以用 R 或 P 表示 R1或 P1. n 在 1 到 64 之间.

MF(演奏)使后面的音乐在"前景"中演奏. 即是，直到演奏完整个乐曲串，没有其它语句被执行. 演奏(MF)与伴音(MB)缺省时，表示是演奏.

MB(伴音)使它后面的乐曲在"背景"中演奏. 即是把音乐串保存在缓冲器里，最多保留 32 个音符. 程序继续运行中反复演奏. 如果少于 32 个音符(或休止符) 可以跟随 MC 命令. 如果超过 32 个音符(或休止符)，多余部分被忽略. 当程序停止时，背景乐曲会自动停止. 如果想在此之前停止一个背景音乐，则需执行另一个带 MF 命令的 PLAY 语句.

MN"正常"地演奏乐曲(与其对应的是"连奏"或"断奏")，每个音符演奏 L 命令指示时间的 7/8，在每个音符后有一点间断，给乐曲一种干净明快的感觉. 演奏方式(MN，ML，MS)缺省，为 MN.

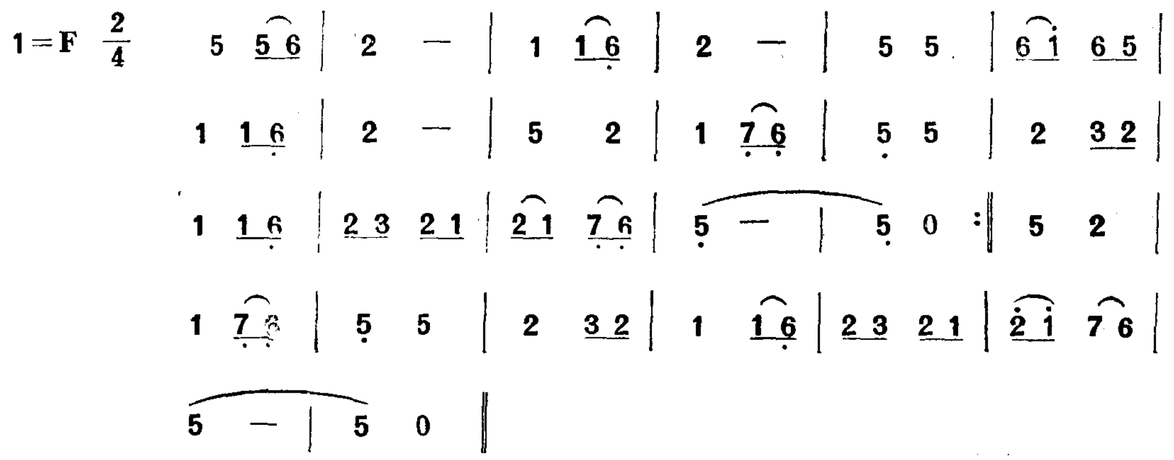
ML"连奏"演奏乐曲，或拉长一点. 每个音符演奏完 L 命令指示的时间，乐曲声音显得较慢，较连绵.

MS"断奏"演奏乐曲，或活泼地演奏乐曲. 每个音符仅演奏 L 命令指示时间的 3/4. 乐曲声音显得很活泼.

空格空格可以放在音乐串的任何所希望的地方，使程序易读，True BASIC 将忽略它们. 字母可以大写或小写. 但是，除上述列出的那些字符以外，其它符号是不准出现在音乐串中的. 下面举例说明如何把一个乐曲翻译成音乐串.

例 1 试编一个程序，演奏"东方红"乐曲.

解 东方红乐曲如下所示:



由于乐曲是 F 调，所以音符最好按如下对应关系翻译:

	O4							O5						
1=F	C	D	E	F	G	A	B	C	D	E	F	G	A	B
	$\underset{\cdot}{5}$	$\underset{\cdot}{6}$	$\underset{\cdot}{7}$	1	2	3	4	5	6	7	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$

按照上述音乐命令规定,选择"连奏"和"演奏"方式,可编程序如下:

```
PROGRAM ex10_1
!
FOR i=1 to 3
DO while more data

    READ music$                 ! Get the string representations
    PLAY music$                 ! And play the notes

LOOP
RESTORE
NEXT !
PLAY "O4 >C4 <G4 F4 E8D8       C4 >C4      <G4 A8G8"
PLAY "O4 F4 F8 D8  L8 GAGF     >GF ED C2   C4 R4"

DATA ML T150
DATA O4 >C4 C8D8    <  G2      F4 F8 D8      G2
DATA    >C4 C4      L8DF DC    <F4 F8 D8     G2
DATA    >C4 <G4     F4 E8D8    C4 >C4        <G4 A8G8
DATA    F4 F8 D8    L8 GA GF   GF ED         C2
DATA    C4 R4
END
```

例 2 试编一个程序,用"嘀哩嘀哩"乐曲为伴音的 Hanoi 塔游戏程序.

解 Hanoi 塔游戏动画程序在第 8 章的 ex08_12a 已作过,这里再加上伴奏.选"断奏"方式,用 C 调,可编程序如下:

```
PROGRAM ex10_2
!
! Towers of Hanoi
!
OPEN #1: screen .1875,.8125,.2,.8
LET disknum = 6
CALL initial(disknum)
!
PLAY "T150 MS"
!
DIM xsize(3,0 to 6)               ! For graphics
LET xsize(1,0) = disknum          ! All on disk 1
```

```
FOR y = 1 todisknum
!   LET xsize(1,y) = disknum+1-y
NEXT y
!
CALL mtower (disknum,1,3,2,xsize)
PLAY "MB O5 L8 EEEC <G4 G8 R8 >EEEC E4 R4"
!
SUB initial(n)
!
    SET WINDOW 0,4,0,8
    SET BACK 17
    SET COLOR 1
    PLOT TEXT, at. 5,10: " The Towers of Hanoi"
!
    SET COLOR 1
    BOX AREA 0,1,0,1
PLAY "MB O5 L8 GGEC <GG C4 AB >CE D4 R4"
    SET COLOR 3
    FOR x= 1 to 3
         BOX AREA x-.02,x+.02,1.05,7
    NEXT x
PLAY "MB O5 L8 GA GA GF EC <G R8 >E R8 DC R4"
    FOR y = 1 to n
         CALL disk(1,y,2,n+1-y)
    NEXT y
RLAY "MB O5 L16FF L8 GFG AAA R8 L16 DD L8 DDD G2"
!
END SUB

SUB mtower(n, s1, s3, s2, xsize(,)) ! Recursive routine
    IF n = 1 then
       CALL mdisk(s1, s3, xsize)
    ELSE
       CALL mtower(n-1, s1, s2, s3, xsize)
PLAY "MB O5 L16CC L8 CCD L8 EEE R8 L16<GG L8GGG >D2"
       CALL mdisk(s1, s3, xsize)
PLAY "MB O5 L8 GA GA GF EC <G R8 >E R8 DC R4"
       CALL mtower(n-1, s2, s3, s1, xsize)
      END IF
END SUB
SUB mdisk(off, put, xsize(,))
```

```
     LET offp = xsize(off, 0)
      LET putp = xsize(put, 0)
      LET   xs = xsize(off, offp)
      CALL disk(off, offp, 0, xs)         ! Erase
      CALL disk(put, putp+1, 2, xs)       ! Draw
      LET       xsize(off, 0) = offp-1
      LET       xsize(put, 0) = putp+1
      LET xsize(put, putp+1) = xs
END SUB

SUB disk(x, y, col, xs)      ! Disk no., heisht, color, size
     LET x1 = x - (xs+1)/15
     LET x2 = x + (xs+1)/15
     LET y1 = y
     LET y2 = 1 + y
     SET COLOR col
     BOX AREA x1, x2, y1+.05, y2
     IF col = 0 then
        SET COLOR 3
        BOX AREA x-.02, x+.02, y1+.05, y2
     END IF
END SUB
END
```

10.2 SOUND 语 句

SOUND 语句可以使计算机发出声音. 仔细地运用 SOUND 语句可以造出各种不同的声音. True BASIC 的 SOUND 语句的一般形式:

SOUND freg, time

其中 freg 和 time 都是数值表达式; freg 给出声音的频率(用赫兹作单位), 而 time 给出了以秒计的持续时间.

执行 SOUND 语句, 发声即开始, 程序继续运行, 即在发声时, 其它语句也被执行. 声音持续到它的时间完结, 或另一个持续时间为 0 的 SOUND 语句被执行, 或程序停止运行. 如果后面的 SOUND 语句持续时间不为 0, 则 True BASIC 等待第一个 SOUND 语句结束, 然后开始第二个 SOUND 语句的声音. 而且再开始执行其它语句. 例如:

```
PROGRAM ex10_3
!  Imitates the sound of a siren.
!
```

```
POR cycle = 1 to 10                          ! Repeat 10 times
!
      FOR freq = 400 to 815 step 15          ! Increasing frequency
            SOUND freq,.02                   ! For .02 seconds
      NEXT freq
!
    FOR freq = 815 to 400 step -15           ! Decreasing frequency
          SOUND freq,.02                     ! For .02 seconds
    NEXT freq
!
NEXT cycle
END
```

第11章　其它程序设计

前面十章,介绍了 True BASIC 的常用语句和模块化结构化程序设计方法，可以满足绝大多数读者工作的需求．本章将介绍 True BASIC 的其它程序设计，供读者进一步学习阅读．对于初学者可以跳过不看．由于篇幅有限,概不作详述,请读者谅解.

11.1　出错与出错处理

在 True BASIC 程序运行时，出错是经常发生的．如果不管它，几乎所有运行态出错都是致命性的.在 True BASIC 中任何运行态出错都可以由出错处理语句来阻止.注意，编译态出错是不可阻止的.

11.1.1　出错处理结构语句

为阻止运行态出错,True BASIC 引入了出错处理结构语句,其形式如下:

```
WHEN ERROR IN
    ......)
    ......}保护块
    ......)
USE
    ......)
    ......}错误处理块
    ......)
END WHEN
```

其中 ERROR 可以用 EXCEPTION(符合美国国家标准)代替.

执行出错处理结构语句,当在执行保护块时,如果没有错误发生，则跳过错误处理块;然而，若在执行保护块时出现了错误，则中断保护块的处理，转入执行错误处理块．当执行到 END WHEN 语句时,则继续执行它后面的语句.

在保护块中，除函数、图画或子程序定义以外，可以设置任何语句，还可嵌套出错处理结构语句.

在错误处理块中,不准嵌套出错处理结构语句．如果函数或子程序在保护块中被调用,那么它的所有语句均受到保护．但是这些函数或子程序还可以有自己的出错处理结构.在这种情况下,被调用程序的出错处理结构是一个子范围,出错处理程序对这些出错处理子范围将给出相应的层次号.

可以使用EXIT HANDLER 语句不进入子范围而返回到高一层去处理.

例如,输入任意一个数,求其倒数,可编程序如下:

```
PROGRAM ex11_1
DO
  WHEN error in
      PRINT "Plase give me one number";
      INPUT n
      LET g=1/n
      PRINT "one over "; n; " is "; g
  USE
     PRINT "Sorry,"; " illegal data."
     PRINT "Plase try again."
END when
LOOP
END
```

运行结果:

```
Plase give me one number? 7↙
one over 7 is .142857
Plase give me one number? 0↙
Sorry,illegal data.
Plase try again.
Plase give me one number? 10↙
one over 10 is .1
Plase give me one number?
Program stopped.
```

运行上述程序可以求任何数的倒数,既使求 0 的倒数,它也会给出"无效的数据"的信息. 程序执行也不中断,还可继续处理.

11.1.2 出错处理函数

True BASIC 为了便于用户编写错误处理块,引入了如下出错处理函数和语句.

(1) extype 函数,其值是一个代码,范围如下:

0	(表示无错.)
1 到 999	(供用户使用,使用方法见后.)
1000 以上	(由 True BASIC 提供信息,见附录 A.2)

(2) extext$ 函数,其值是一个字符串,形式如下:

空串	(表示无错.)
错误信息	(表示出错原因.)

(3) exline$ 函数,其值是一个字符串,形式如下:

空串	(表示无错误.)

行号 in 程序名 (表示在指定程序的指定行出现了错误.)

系统规定 Main 表示主程序名.

这里仍以求倒数为例. 如下程序在保护块中调用了一个子程序.

```
PROGRAM ex11_2
DO
  WHEN exception in
      PRINT "Plase give me one number";
      INPUT n
      CALL badsub(n,g)
      PRINT "one over "; n; " is "; g
  USE
     PRINT extype;" "; extext$
     PRINT exline$
  END when
LOOP
END
SUB badsub(n,g)
  LET g=1/n
END sub
```

运行结果:

```
Plase give me one number? 9 ↵
one over 9 is .111111
Plase give me one number?  7 ↵
one over7 is .142857
Plase give me one number? 0 ↵
 3001 Division by zero.
15 in badsub,6 in Main
Plase give me one number? 10 ↵
one over 10 is .1
Plase give me one number?
Program stopped.
```

在程序的错误处理块中,使用了 extype,extext$ 和 exline$ 函数. 当输入数 n 为 0 时,运行给出了相应的输出.

11.1.3 设置出错代码和信息语句

除系统规定的出错信息和代码外,还可以使用CAUSE ERROR 语句, 定义出错代码和出错信息. 其形式如下:

CAUSE ERROR expr,expr$

其中,expr 为数值表达式, 取整用来表示出错代码. 当定义的出错代码与True BASIC 中定义的代码相同时, 一定会发生同样的错误.由extype 函数返回. 出错代码为1 到 999 之间的任意

整数时，是供用户直接使用的，True BASIC 系统没有使用它们. 但用户并非一定只能用这些值，而是可以使用任何的出错代码.

expr$ 为字符表达式，用它设置出错信息，由 EXTEXT 返回.

例如. 模拟计算器程序.

```
PROGRAM ex11_3
! Simple calculator
! Commands are "+n" and "-n"
!
PRINT "Calculator! "
LET n=0
DO
    LINE INPUT cmd$
  IF cmd$="end" then exit$ do
  LET cmd$=Trim$(cmd$)
  LET arg$=cmd$[2:maxnum]
  LET  cmd$=cmd$[1:1]
  WHEN error in
      SELECT CASE cmd$
         CASE "+"
          CALL plus(n,arg$)
         CASE "-"
          CALL minus(n,arg$)
         CASE else
          CALL error(cmd$ & " isn't a command.")
      END SELECT
      PRINT "Total="; n
  USE
      IF extype < > 666 then
      PRINT extype; extext$
      ELSE
       PRINT cmd$ & arg$;"is illegal data"
      END IF
  END WHEN
LOOP
END
!
! Plus. add a number.
!
SUB Plus(n,arg$)
```

```
    DECLARE def myval
    LET n=n+myval(arg$)
END sub
!
! Minus. Subtract a number.
!
SUB Minus(n,arg$)
    DECLARE def myval
    LET n=n-myval(arg$)
END sub
!
DEF myval(s$)
    WHEN error in
        LET myval=val(s$)
    USE
        EXIT HANDLER
    END when
END def
!
SUB error(mess$)
    PRINT "Sorry, " & mess$
    CAUSE ERROR 666,"Tr  - or +"
END sub
```

运行结果:

```
Calculator !
? +66666 ↵
Total=66666
? -555 ↵
Total=66111
? -800 ↵
  4001 VAL string isn' t a proper number.
?   100 ↵
Sorry, 1 isn't a command.
  666 Try-or+
? end ↵
```

用计算器作加法或减法运算时,可能出现两类错误:一类是运算符错,另一类是数值错.在上述程序运行中,若运算符出错,则自动调用子程序error处理,它打印输入字符"是不能执行的命令",并定义错误代码666和错误信息"Try-or+".在上述程序运行中,若数值出错,会在调用函数myval时出错,则执行EXIT HANDLER语句,转入高一层处理,即转入主程序错误处理块.这时出错代码不等于666,打印出错代码和信息,这些是系统给出的(如,输入-800).

11.2 DO 程 序

True BASIC 系统提供了 FORMAT, PAGE, NUM, UNNUM 和 RENUM 五个 DO 程序. 这些程序都属于软件工具，用 DO 命令可以执行. 如键入 DO FORMAT，即可执行 FORMAT程序，把当前文件书写格式化. 这样的 DO 命令给用户编程带来了很大方便. 为满足用户不同的需要，True BASIC 允许用户编写自己需要的 DO 程序，形成新的 DO 命令，用以检查或修改当前文件. 本节将说明怎样形成新的 DO 命令.

11.2.1 DO 程序的建立

DO 程序与其它程序稍有不同，一个 DO 程序需用一个特定的外部子程序代替主程序. 这个外部子程序必须有 2 个参数：前边是一个串数组，后边是一个串变量. 串数组与当前文件(即被处理的源程序)结合，每个数组元素装一行；串变量将与 DO 命令行的选择项结合. 如果缺省选择项则该串为空串；串数组的下限总为 1，上限是可以改变的，主要取决于当前文件有多少行. 因此 DO 程序允许使用 Ubound 找出当前文件的行数.

DO 程序中可有多个外部过程，True BASIC 总是把第 1 个外部过程作为"主过程".

例 把当前文件改为用小写字母书写的文件，并打印出来.

解 按照上述规定可编 DO 程序如下：

```
 ! PROGRAM ex11_4
! DO program which changes the current file
! entirely to lowercase.
!
EXTERNAL
SUB lower(line$( ),arg$)
   FOR i=1 to ubound(line$)
      LET line$(i) =lcase$(line$(i))
      PRINT line$(i)
   NEXT i
END sub
```

在 DO 程序中，通过修改数组元素内容改变当前文件，如果需要，可以更大范围地改变当前文件. 如果要使当前文件长些或短些，可以用 MAT 语句调整串数组的大小.

11.2.2 DO 程序的编译和存盘

在写好一个 DO 程序且调试正确之后，必须编译它并存入磁盘，然后才能用 DO 命令使用它. 注意，未编译过的 DO 程序不能用 DO 命令使用它.

使用一个坏的 DO 命令，将使当前文件受到破坏，因而使用新的 DO 命令时要特别慎重. 譬如，把当前文件存盘留下一个副本；认真调试 DO 程序等等. 以防造成难以挽回的损失.

正确的操作过程如下:

```
OK. save ex11_4
  12 lines saved. (生成 ex11_4.TRU 文件)
OK. compile
  12 lines compiled.
OK. save ex11_4(生成 ex11_4. TRC 文件)
```

这样，就形成了 ex11.4 DO 命令. 利用它可以把任何正文文件的所有字母变为小写字母. 下面是对 ex11_4 程序(文件)的处理操作过程和结果.

```
OK. old ex11_4                调入 ex11_4. TRU 文件
OK. do ex11_4                 运行 ex11_4(TRC)DO 命令.
! program ex11_4
! do program which changes the current file
! entirely to lowercase.
!
external
sub lower(line$( ),arg$)
  for i=1 to ubound(line$)
    let line$(i)=lcase$(line$(i))
    print line$(i)
  next i
end sub
```

11.3 压缩辅程序

True BASIC 提供了两个压缩辅程序,Packb 和 Unpackb,利用它们可以实现把一个整数压缩到串中. 这两个辅程序是供有经验的用户使用的高级功能. 对于初学者,可以跳过不读.

11.3.1 Packb 子程序

在程序中调用形式为:

Call Packb(s$,bstart,blen,value)

其中 s$ 为串变量.

Value 值为十进位数,存储时自动四舍五入取整,再化为二进制位字段.

bstart 值为整数. 它是 Value 值的位字段在 s$ 中的起始位.

blen 值为整数. 它是 Value 值的位字段长度. 执行 Packb 子程序，是把 Value 值压缩到串 s$ 中去,Value 值被四舍五入取整,存储是由串的起始位算起偏移 bstart 位开始，在 blen 长的位字段中.

例如:

Packb(s$,1,8,12)(合并数 12 到串 s$ 的第 1 个字节)

Packb(s$,1,16,12)(合并数 12 到串 s$ 头 2 个字节)

Packb(s$,33,1,0)(合并数 0 到串 s$ 的第 33 位)

串中第 1 个二进制位的位置是 1. 如果 bstart 小于 1, True BASIC 用 1 代替. 如果 bstart 超过串尾,True BASIC 用串尾代替. 如果该位字段置于串尾,串将被自动延长. 如果 Value 值的位字段太长,则只有最右边的 blen 位被存储,多余的高位被废弃.

11.3.2 Unpackb 函数

在表达式中调用形式为

Unpackb (s$,bstart,blen)

其中,bstart 是一个整数的位字段在 s$ 中的起始位. blen 是一个整数的位字段的长度.

调用函数 Unpackb,则从一个串中拆出一个位字段,并把它转换为一个十进制整数.

如果 bstart 小于 1 就用 1 取代;如果 blen 大于串长度,就用串的长度代替.如果位字段扩展出了串的尾部,那些不存在的位就作为 0 处理.

如果 blen 是正数,位字段作为无符号整数处理. 这样,拆出的值在 0 和 2^{blen-1} 之间. 如果 blen 是负数,它的绝对值作为位字段长度,并且拆出的值作为带符号整数处理,其值在 -2^{blen-1} 和 $2^{blen-1}-1$ 之间.

例如:

Uupackb(s$,1,8)返回第 1 个字节,其值为在 0 至 255 之间的整数.

Unpackb(s$,33,1)返回第 33 位,其值为 0 或 1.

Unpackb(s$,1,-16)返回头二个字节,其值为在 -32768 至 32767 间的整数.

11.3.3 压缩子程序的示例

当有大量数据需要保存在存储器中时,压缩程序十分有用. 例如,如果有一大批值为 0 或 1 的数,把它们压缩到一个串中,比把它们保存在数组中要少占用很多存储单元.

例如,找出不大于 100 的素数. 这里使用 Evatoshenes 筛法,在素数倍数(非素数)的位置设置为 1,素数位置保持为 0.

```
PR OGRAM ex11_5
!Prime number sieve Packs the sievearray as bits.
!
LET n=0
LET upper=100
LET s$=repeat$(chr$(0), (upper+7/8))
FOR i=2 to upper step 1
    IF unpackb(s$, i, 1) =0 then
      LET n=n+1
      IF mod(n,5)< >0 then
        PRINT using '####': i;
      ELSE
        PRINT using '####': i
```

```
        END if
        IF i*i <upper then
         FOR j=2*i to upper step i
            CALL packb(s$, j, 1, 1)
         NEXT j
        END if
      END if
    NEXT i
    END
```

运行结果:

2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71
73	79	83	89	97

11.4 链接程序

让几个程序一个接着一个执行的功能,对于用微机处理较大任务来说很有实用价值.为此,True BASIC 引入了 CHAIN 语句.当一个 True BASIC 的 CHAIN 语句执行时,当前程序被停止,然后转去执行它指定的程序.CHAIN 语句与 PROGRAM 语句要配合使用.

11.4.1 链接程序形式之一

CHAIN expr$ [,RETURN]

其中 expr$ 是被链接程序的文件名,程序不带参数.方括号内的项表示是可选项.

在 CHAIN 语句执行时,除显示屏/键盘通道以外,所有通道都被关闭,并释放全部变量的存储单元.如果希望把当前程序与其它程序链接,并且在被链接程序执行完成后返回到当前的程序,这时需要选择 RETURN 项,系统将保留当前程序所用的存储单元.

例如,下面的程序是先把一个新的 True BASIC 程序写到磁盘上,然后链接它,当新的程序执行完后,继续执行原来的程序.

```
PROGRAM ex11_6
OPEN #2:printer
OPEN #1:name "temp",access outin,organization byte
ERASE #1
PRINT #1:"OPEN #4:printer"
PRINT #1:"PRINT #4:1+5"
PRINT #1:"END"
CLOSE #1
CHAIN "temp",return
```

```
PRINT #2:" All done."
END
```

运行结果:

```
6
    All done.
```

同时在磁盘上生成如下一个新的 temp. tru 文件:

```
OPEN #4:printer
PRINT #4:1+5
END
```

11.4.2 链接程序形式之二

被链接的程序可以带虚拟参数,然后用链接语句赋予实在参数:

CHAIN expr$ WITH (arg$) [,RETURN]

PROGRAM name(var$)

其中,expr$ 的值是程序 name 的文件名.

var$ 为虚拟参数.

arg$ 为实在参数.

可选项 RETURN 的含义和上述链接程序形式之一所述一致.

例如,如下程序是传送一个字符串(日期函数 date$ 的函数值)到被链接的程序中.

```
PROGRAM ex11_7
! Demonstrate passing paramter on CHAIN
!
LET example$=date$
CHAIN "fname" with(example$)
ENd
```

运行上述程序时,在磁盘上应有文件名为 fname 的程序,如下所示.

```
! chain ex11_7
PROGRAM fname(string$)
PRINT string$
END
```

运行结果:

```
19880110
```

11.5 汇编语言程序的调用

用汇编语言编写的函数和子程序作为外部辅程序可以在 True BASIC 程序中使用. 和调

用 True BASIC 外部辅程序类似，在调用之前，应加上一条 LIBRARY 语句．不同的是汇编语言函数的参数传递要靠一张引用表．这样，又带来一些新问题，这对于能使用汇编语言编写程序的读者是极容易的．由于篇幅有限，这里不作详细介绍，请读者参考 True BASIC 使用说明的附录 E.

第12　章上机操作方法

学习计算机语言程序设计，目的是为了编写应用程序，用以控制计算机处理数据．最好的学习方法是把理论与实践结合起来．

实践是检验真理的唯一标准，程序的对错，效率的高低，要靠上机检查．数据处理结果要通过上机才能得到．另外，在机器上调试程序（排错）也是程序设计能力的重要组成部分．所以，本书把上机操作方法作为本课程的一部分给予介绍．

上机实验要受到设备条件的约束，无法强求一致．本章内容可酌情有选择地插到第 2 章到第 7 章之间学习．在条件允许的情况下，程序设计上机实验时间应是讲课学时的 1/3 到 1/2，另外再抽几天时间作个课程设计．

下面以IBM PC机为例，介绍一下有关 True BASIC 的上机操作方法．这些方法对于 IBM PC/XT 等高档机，或兼容机，如 super 微型机、长城 0520 微机都适用，或基本适用．

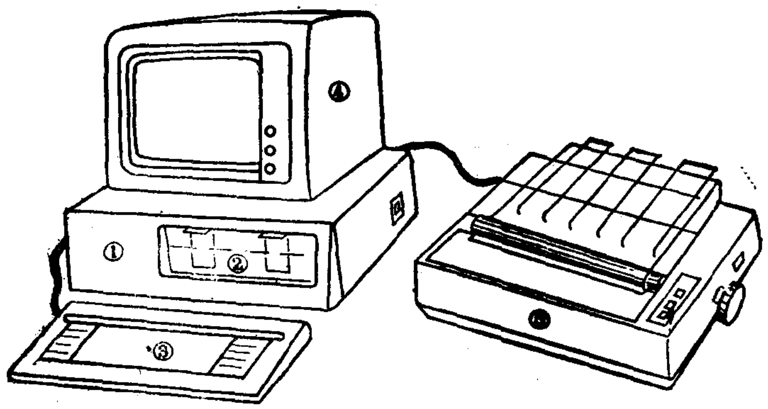
12.1　IBM PC系统简介

本节简要地介绍一下 IBM PC 机系统的组成、开机与关机以及操作系统（DOS）常用命令．

在第 1 章曾对计算机的组成作过一般地介绍．IBM 个人计算机（PC）系统也分两大部分，即硬件和软件．

12.1.1 硬件．

IBM PC 系统硬件配置如图 12-1 所示．



① 主机　② 驱动器　③ 键盘　④ 显示器　⑤ 打印机

图 12-1　IBM PC 系统硬件配置

0328391

（1）主机：主机由运算器、控制器和存储器组成．它的内存储容量为 256 K 或 512 K．

（2）驱动器：IBM PC 系统具有两个 5.25 英寸的软盘驱动器，左边叫 A 驱动器，右边叫 B 驱动器．当显示 A>时，表示 A 驱动器正和主机连接着，这时也称驱动器 A 为当前驱动器．当它的指示灯亮（红）时，表示正在从软盘上存取数据．如果键入

A>b: ↵　　（↵表示回车键）

则显示

B>

这时表示驱动器 B 为当前驱动器．

软盘，它是一种磁性存储器，也叫软磁盘，放在主机之外，它是一种外存储器．目前一块双面双密度软盘存储总容量为 360 K．软盘是放在盘套中的，如图 12-2 所示．

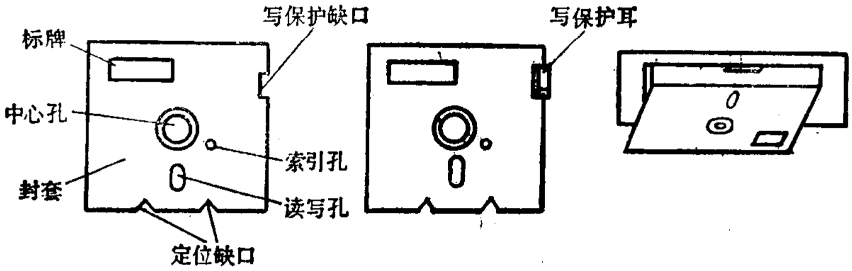


图　12-2

插在驱动器 A 中的软盘简称 A 盘，插在驱动器 B 中的简称 B 盘．软盘必须在驱动器停止转动时（即指示灯不亮时）插入或取出，否则会损坏软盘和驱动器．插入软盘时，软盘的定位缺口向前，带有商标的一面向上，插入后要关好驱动器的门．

（3）键盘：键盘如图 12-3 所示．

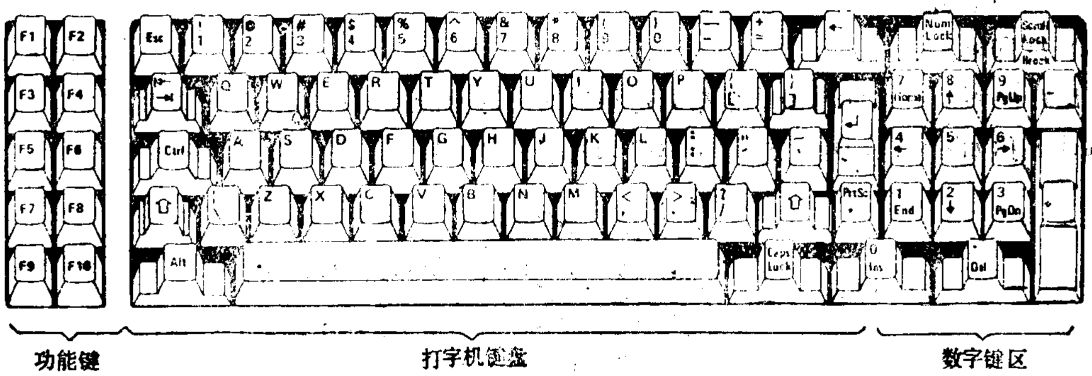


图　12-3

共有 83 个键，分三个区．左边 F1～F10 为功能键区，供各系统软件特殊定义使用．右边为数字键或光标控制键区．中间为数据键区．边缘为控制键．常用键如下：

数据键：Aa～Zz，0～9，特殊字符．空格键．

控制键：

↑	移位键，有的键盘它的形状是 shift ，控制双挡键的如 ↑ − ± 表示“＋”.
Caps Lock	它是控制字母大小写的开关，按一下大写，再按一下变小写.
Num Lock	它是控制它下面 9 个数字键的开关，按一下数字有效，再按一下，控制光标移动键有效.
↵	回车键， 有的键盘它的形状是 Return ，如键入一个命令，用该键送入执行.
Ctrl + Break	叫 STOP 键，仃止当前命令的执行.
↑ + PrtSc	叫硬拷贝，打印屏幕上的内容.
Ctrl + Alt + Del	叫热启动，在接通电源的条件下，重新启动.
←	抹去光标前面的字符.
4 → 6 ← 8 ↑ 2 ↓	为光标向右、向左、向上、向下控制键.

其它键的功能等以后再讲.

（4）显示器.

显示器有一个指示灯和两个旋钮：一个为亮度旋钮；另一个为对比度旋钮. 通常能显示 25 行，每行 80 列字符.

（5）打印机.

打印机的型号有多种，但结构布局是相似的. 这里对程序员关心的问题作个简介. 打印机部分部件名称如图 12-4 所示(这是拿掉打印机盖后的视图.)

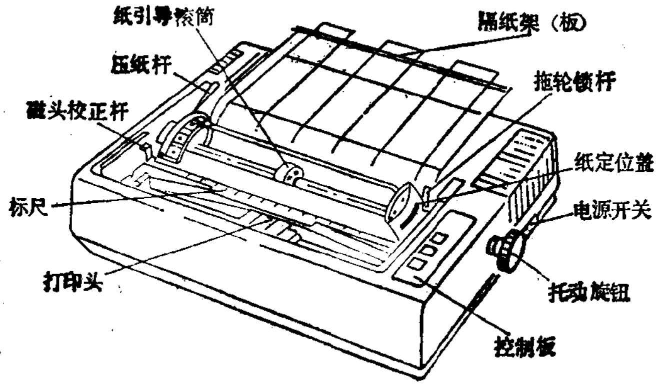


图 12-4

a) 纸的安装.

打印机拖动纸有两种方式，一种用压纸杆压纸拖动方式；另一种用拖动轮拖动方式. 纸从隔纸架下面，从上到下引入，经过拖动轮下部绕到上面. 要把纸从下部送到上部，若放松压纸杆，可用手慢慢送过来；若压紧压纸杆，可用拖动旋钮将纸送过来，松开拖轮锁杆，把拖轮放到合适位置，把纸放入拖轮，盖好定位盖，锁上拖轮锁杆，压上标尺，盖好机盖. 这就是拖动轮拖动方式. 如果纸不经过拖动轮，直接从它上面通过，这时要把压纸杆压紧，这就是压纸杆压纸托动方式. 一般地不要两种方式同时用，以免相互干扰.

b) 打印机控制板.

控制板有四个指示灯:

Power　　当打印机接通电源时亮（要打开打印机电源开关）.

Ready　　当打印机准备好可打印数据时亮（要安装好纸）.

Paper Out　当纸接近用完时亮（告诉程序员准备换纸）

On Line　　当打印机处于在线模式时亮（表示和主机连接好了）.

控制板有三个开关:

On Lin　连接主机的开关.

FF　　走纸开关(Form Feed)，当打印机不在线时，可按该开关，连续走纸.

LF　　换行开关(Line Feed)，按一下走一行.

c) 打印宽度.

每行打印 80 列或 132 列. 可用如下操作系统命令重新设置.

A>mode LPT1: n, m　　(n 为 80 或 132, m 为 6 或 8)

m 表示垂直方向每英寸的行数，用以控制行的疏密

12.1.2 软件

IBM PC 机软件有操作系统和 FORTRAN, PASCAL, True BASIC 等高级语言的编译系统

在磁盘操作系统(DOS)状态下，可以用 DOS 命令处理文件，如打印文件、拷贝文件等. 文件标识符形式如下.

文件名.后缀

其中，文件名通常是字母开头，由 1 到 8 个字母、数字 下划线 _ 组成.

后缀表示文件的属性，FOR 表示是FORTRAN 源程序文件，PAS 表示是 PASCAL源程序文件，BAS 表示是BASIC源程序文件，TRU 表示是 True BASIC 源程序文件，TRC 表示是 True BASIC 程序编译后的目标文件，等等.

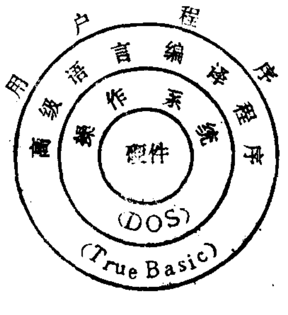


图 12-5

在磁盘操作系统支持下，可以运行各种高级语言的编译系统. True BASIC 软件要在 DOS 2.0（第 2 版本）或更高的版本支持下运行，如图 12-5 所示.

12.1.3 开机与关机

(1) 开机。

将DOS盘插入A驱动器(左边的)，关上驱动器门。接通显示器和PC机的电源。这时机器首先自检内存，然后驱动器红灯亮，表示正从软盘上读入系统命令到内存。短时间过后，显示器上显示一些文字，计算机在问开机的日期(date)和时间(time)。为了简单，这时可以连按二次回车键(↙)。如下所示:

Current date is Tue 1—01—80

Enter new date: ↙

Current time is 0:00:10.54

Enter new time: ↙

The IBM Personal Computer DOS

Version 2.00(c) Copyright IBM Corp 1981,1982,1983

A>—

这就是磁盘操作系统DOS的提示符。这时计算机处于待命状态，可以输入任何DOS系统命令。命令用大写或小写字母都一样，只要字符(包括空格、标点符号)准确无误即可。回车键是命令或书写行的结束符号。在没有键入回车键之前，可以反复修改命令。键入坏的命令(bad Command)是不会损坏机器的，计算机会重新给一个提示，A>—，就可重新输入命令。

(2) 关机。

当计算机处于DOS提示符状态时，可以取出软盘，依次关闭显示器、打印机和主机即可。

12.1.4 DOS系统常用命令

DOS命令分内部命令和外部命令两大类。内部命令在COMMAND·COM文件中。外部命令作为独立文件存放在DOS盘上。

(1) COPY命令(拷贝命令，内部)。把一个或多个文件复制到一个磁盘上。如:

A>COPY 源文件标识符，B:[目标文件标识符] (把源文件从A盘复制到B盘上，若目标文件标识符缺省，则为同名文件)。

A>COPY 源文件标识符，目标文件标识符 (把源文件从A盘复制到A盘上目标文件名下)。

A>COPY *·*，B: (A盘全部文件拷贝到B盘上。注意，不拷贝子目录内的文件。)

(2) DISKCOPY命令(全盘拷贝命令，外部)。如:

A>DISKCOPY A: B:

DISKCOPY是把A盘内容全部复制到B盘(目标软盘)上。在复制期间，如果必要，可以格式化目标软盘。复制之后，提示COPY another(Y/N)? 如果有其它盘要复制，可以回答Y，否则回答N，结束。

(3) DIR命令(查目录命令，内部)。

A>DIR /w (横向列出所有的目录项。)

A>DIR（纵向列出所有的目录项．并提示盘上剩余空间数．）

A>DIR *·TRU（列出所有后缀为 TRU 的文件名．）

A>DIR ex 01*·*（列出所有以 ex 01 为开头的一切文件名．）

(4) **DEL 命令**（删除命令，内部）．

A>DEL 文件标识符 （删除指定文件．）

A>DEL *·TRC （删除所有后缀为 TRC 的文件．）

A>DEL *·* （删除盘上所有文件，注意不要乱用．）

A>DEL ex 01*·* （删除所有以 ex 01 为开头的文件．）

(5) **REN 命令**（修改文件名命令，内部）．

A>REN 旧文件标识符，新文件标识符

(6) **TYPE 命令**（打印命令，内部）．

A>TYPE 文件标识符 （打印显示指定文件．）

(7) **FORMAT 命令**（格式化命令，外部）．

A>format b:/v

系统显示出下列信息：

Insert new diskette for drive B:
and strike ENTER when ready

用户插入新的软盘，并打入任一键之后，系统显示如下信息：

Formatting...Format complete
Volume label (11 characters, ENTER for none)? TBASIC ↵
    362496 bytes total disk space　　（全部磁盘空间）
    362496 bytes available on disk　　（用户可使用的空间）
Format another (Y/N)?n

在上例中输入 TBASIC 作为盘标记．如果在 FORMAT 命令中不带/v，就没有输入盘标记的提示．若加/s，则会把操作系统文件按下列顺序从当前驱动器中复制到新的软盘上：

IBMBIO·COM （基本 I/O 处理程序，隐藏文件，2K）

IBMDOS·COM （磁盘文件管理程序，隐藏文件，6.5 K）

COMMAND·COM （命令处理程序，5 K）

(8) **AUTOEXEC·BAT 文件**（专用批命令文件，外部）．

当用户启动或重新启动 DOS 时，命令处理程序自动检索 AUTOEXEC·BAT 文件．如果该文件在 DOS 所启动驱动器的根目录中，每当用户启动 DOS 时，DOS 将自动地执行该文件．

如果用户想自动地装入 True BASIC 系统，则应按如下步骤产生 AUTOEXEC·BAT 文件：

① 输入

A>COPY CON:，AUTOEXEC·BAT

该命令告诉 DOS 把来自键盘(CON)的信息复制到 A 盘的 AUTOEXEC·BAT 文件中．

② 再输入

a:Hello　　(Hello 为 True BASIC 软件文件名)

按F6结束从键盘到 AUTOEXEC·BAT 文件的操作. 然后，把 AUTOEXEC·BAT 文件复制到 True BASIC 盘上. 如此,机器启动后就会自动进入 True BASIC 系统状态.

(9) **GRAFTABL 命令**(装入图表命令,外部).

A>GRAFTABL↙

显示: graphics characters loaded

这时,系统处在图形方式下,将支持 ASCII 表第 128 到 254 字符的显示.

(10) **GRAPHICS 命令**(屏幕打印命令,外部).

A>GRAPHICS↙

输入该命令后，再按↑+PrtSc键可打印屏幕上显示的图形.

12.2 实验前的准备*

在使用 True BASIC 软件之前,要先做两件事:第一,对原版的 True BASIC 软盘要复制一个备份盘,并将原版软盘保存在阴凉、干燥的地方,以备复制盘受到损坏时，再复制新盘. 第二,把DOS 系统拷贝到新盘上. 这样使用起来比较方便.

在双软盘驱动器上,具体操作方法如下:

12.2.1 复制 True BASIC 软盘

开机后,键入 diskcopy 命令

A>diskcopy a:,b:↙

显示: Insert Source diskette in drive A:

Insert target diskette in drive B:

Strike any key when ready

意思是,在左边的 A 驱动器中插入源盘(这里是 True BASIC 原版盘),在右边的 B 驱动器中插入目标盘(这里是一个空白盘). 关上两个驱动器门，按任何一键，计算机开始复制工作. 直到显示:

Copy Compllte (拷贝完成)

Copy another (Y/N)?n↙

注意:如果用 COPY 命令

A>COPY *.*, b:↙

这时会丢失 TBHELP 子目录的文件,即丢失了"帮助"功能.

12.2.2 把 DOS 复制到新拷贝的 True BASIC 软盘上

把 DOS 软盘放入 A 驱动器,把新拷贝的 True BASIC 盘放入 B驱动器. 然后键入命令

A>b: INSTALL↵

显示: A>a:

A>sys b:

system transferred

再键入:

A>COPY Command·Com, b: ↵

1 File(s) Copied

显示: A>

经过 12.2.1 和12.2.2 的操作，一张新的 True BASIC 盘已复制完. 这时，请验证一下，复制是否成功. 可把新 True BASIC 盘放入 A 驱动器，重新热启动，即按 [Alt]+[Del]+[Ctrl]；或关掉电源，过一段时间再加电重新开始. 如果复制 DOS 是正确的，大约一分钟之后屏幕上又会出现日期和时间的提示. 这时一切处理和用 DOS 盘启动一样，不再重述.

12.2.3 True BASIC 系统主要程序

COMMAND	COM	17664	命令管理系统
HELLO	EXE	99568	编译系统
INSTALL	BAT	33	初始化程序
TBHELP	〈DIR〉		帮助子目录
PAGE	TRU	2910	分页打印命令
FORMAT	TRU	8647	格式化 DO 命令
UNNUM	TRU	452	删行号 DO 命令
NUM	TRU	1258	添行号 DO 命令
RENUM	TRU	4463	重编行号 DO 命令
FNMLIB	TRC	1424	数字函数库
I HLTB	TRC	1529	双曲线函数库
FNTLIB	TRC	977	三角函数库(以弧度制)
FNTDLIB	TRC	1055	三角函数库(以角度制)
GRAPHLIB	TRU	4850	图形函数库
MENULIB	TRU	4669	菜单子程序库

在进入 DOS 状态后，只要有 HELLO·EXE 程序即可. 其它程序根据工作需要自由选用. 比如在编程过程中，不使用外部程序库，就可以从盘上将它们删去. 这样盘上就可以有更多空间供用户使用. 更多的盘空间对组织学生实验是很需要的.

12.3 True BASIC 的进入与退出

12.3.1 进入 True BASIC

当开机之后，显示器显示 A>，把 True BASIC 盘放入 A 驱动器，并键入 HELLO:

A>HELLO↲

稍等片刻，在屏幕底部显示如下:

True BASIC here

Version 1.0

Copyrighe(c) 1985 by True BASIC, Inc.

Published by Addison-Wesley Publishing Company, Inc.

OK.

“OK”是 True BASIC 的提示符. 在这儿可以键入 True BASIC 命令.

12.3.2 屏幕上的窗口

在进入 True BASIC 状态之后，屏幕分为上下两个区域，上面的区域叫编辑窗口 (editing window)，供用户编辑自己的 True BASIC 程序用. 下面的区域叫背景窗口 (history window)，供用户输入 True BASIC 命令，或显示程序运行结果. 由于它能给出人机对话期间的痕迹，故取名背景窗口. 如图 12-6 所示.

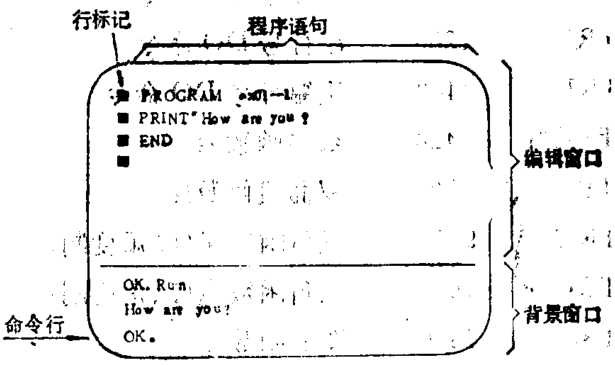


图 12-6 True BASIC的窗口

(1) 窗口转换.

两个窗口，一个可以输入 True BASIC 命令，一个可以编辑用户程序，光标可以在两个窗口上转换. 按F1键，光标移至编辑窗口；按F2键，光标移至背景窗口，而且仍回到它离开时的位置.

(2) 窗口大小.

编辑窗口系统既定值是 17 行，背景窗口是 7 行. 中间横线占 1 行. 利用如下命令，可以重新定义编辑窗口大小:

OK. split n　　(0≤n≤24)

表示编辑窗口为 n 行. 在硬拷贝之前常用它扩大背景窗口,以便打印出所需结果.

(3) 窗口的前景与后景颜色.

如果屏幕是彩色的, 就可以使用 Ctrl 键和字符 F,B 及 E 来改变屏幕的颜色.

·光标在编辑窗口,按 Ctrl +F 键改变编辑窗口前景颜色. 按 Ctrl +B 键改变编辑窗口后景颜色. 若连续按动,则连续改变(循环).

·光标在背景窗口,按 Ctrl +F 键改变背景窗口前景颜色. 按 Ctrl +B 键改变背景窗口后景颜色. 若连续按动,则连续改变(循环).

· 按 Ctrl +E 键是改变屏幕边缘颜色. 若连续按动,颜色可以连续改变,可选择任一喜欢的颜色.

12.3.3 退出 True BASIC

退出 True BASIC 可键入如下命令:

OK. bye

显示　A>-

表示机器已退出了 True BASIC 回到 DOS 状态. 此时,可以把 True BASIC 盘从驱动器中取出. 关机结束.

12.4 常用 True BASIC 文件处理命令和程序控制命令

True BASIC 系统命令,可分为三类:编辑命令(见 12.5 和12.6), 文件处理命令和程序控制命令,如表 12-1 所示.

表 12-1

命　令	参　数	功　能
New		清除内存,光标移到编辑窗口起点.
Run (或 F9)		解释,运行内存中的程序(也叫当前文件).
Svae	文件名	把当前文件登目,并存入文件名. TRU名下. 如果已存在同名文件,显示出错信息,内存不变.
Unsvae	文件名	删除盘上指定文件. 文件名可用类似于DOS的DEL命令参数(见12.1.4(4)).
Old	文件名	清除内存,调指定文件进入内存,并显示在屏幕上.
Replace	文件名	删除软盘上指定文件,并将当前文件存入该名下. 如果盘有写保护,结果可以删除,但不能写入,显示出错. 内存不变.
List	[参数]	缺省打印当前文件,内存不变. 参数: L_1--L_2或模块名. 打印指定行或模块.
File	[参数]	显示True BASIC盘上文件目录. 参数与DOS的DIR命令类似(见12.1.4(3)).

命令	参数	功能
Help 或 F10	参数	True BASIC对一些问题可提供屏幕解释. 这时屏幕上两个窗口被清除. 显示提供的解释清单. 当按任一键时,将返回到清屏前的状态. 如参数为topics,显示所有能获得帮助的参数清单. 这些解释都存在盘上TBHELP子目录中.
Do format		把当前文件书写格式化,是缩进式的.
Compeil		把当前文件编译变成编译代码. 注意,这时当前的源程序文件消失了. 如果还打算保留源程序文件,应在编译前做好更名复制工作. 编译代码比源程序代码条数少得多,运行也快的多. 可以商品化. 存盘后,文件标识符后缀为. TRC.
Do num		给当前文件添加行号.
Do Unnum		把当前文件行号删除.
Do Renum	起行号,增量	把当前文件重编行号.
Do page		分页打印当前文件.

12.5 初级编辑

学习本书的前三章之后,读者可以编写一个顺序程序,然后用键盘逐字逐句地送入内存、调试、运行,直到存入磁盘.

12.5.1 程序(文件)的编辑

如果要编写一个新程序,键入如下 True BASIC 命令.

OK. new

编辑窗口清屏,光标移至编辑窗口的第 1 行第 1 列. 然后,逐字逐句地键入程序.

如果程序是软盘上的一个旧文件,键入如下 True BASIC 命令:

OK. old 文件名(文件名不用带. Tru)

指定文件调入内存,并显示在屏幕上.

True BASIC 编辑是一个全屏幕编辑,即不受行和列的限制,光标可以通过如下控制键移到程序的任何位置:

光标移动键:

4 ←	左移一个字符	8 ↑	上移一行
6 →	右移一个字符	2 ↓	下移一行
Home	移至窗口顶部(程序第 1 行)	PgUp	向上翻页
End	移至窗口底部(程序最末行)	PgDn	向下翻页

⇄	左移一个字①	Ctrl – 4 ←	移至行首
↑ – ⇄	右移一个字	Ctrl – 6 →	移至行尾

然后可利用如下编辑键来作字符编辑:

←	删除光标左边一个字符
Del	光标在任一字符上,删除当前字符
Del	光标在行标记上,删除当前行
Ctrl - Home	删除左边一个字
Ctrl - PgUp	删除当前一个字
Esc	删除光标左边行部份
Ctrl - End	删除光标右边行部份
F7	恢复最后的插入或删除
Ins	为插入/置换开关. 光标显示"▌"时,为置换状态,用键入字符置换当前字符;光标显示"_"时,为插入状态,键入字符插入当前字符位置,所有右边字符右移.

程序输入、修改之后,按 F2 移光标至背景窗口,再作文件处理,如运行、格式化、存储等等.

12.5.2 True BASIC 命令的编辑

命令编辑为行编辑. 已执行过的命令是不能修改的. 在当前行上,和在编辑窗口一样,可以移动光标,以及插入、删除、修改字符等. 此外,还可以用 F6 拷贝已执行过命令的全部或一部分到当前行. 方法是,把光标移到已执行的命令行,再按 F6,这时光标右边部分将拷贝到当前行光标离开时的左边,光标停在末尾. 如

OK Old ex05_4

OK Run

OK Replace_

这时,将光标移到 ex05_4 的 e 处,按 F6,则 ex05_4 被拷贝到原光标左边. 即形成如下命令:

OK Replace ex05_4

这样,既快又可减少差错.

① 字:前后用分隔符(如,空格,·,:,|,=等)分隔的字符串.

12.6 高级编辑

学会使用F3～F6功能键来编程序，包括在程序中查找指定的字，用新字置换旧字，以及学会删除、复制、前后或左右移动，编辑行模块等.

True BASIC 系统中的功能键功能如下：

F1	移光标到编辑窗口
F2	移光标到背景窗口
F3	Find(查找)命令
F4	行标记明/暗开关
F5	Copy (拷贝)命令
F6	Move(移动)命令
F7	Restore (恢复)命令. 恢复最后的插入或删除
F8	空缺，未定义
F9	Run(运行)命令
F10	Help(帮助)命令，可获得系统提示，其中除F3～F6以外，各键功能都已讲过.

12.6.1 Find(查找)命令

用该命令可在用户程序中查找出指定的字，操作方法如下：

(1) 把光标移至编辑窗口上查找的起点，即用光标指定查找起点.

(2) 按F3键，这时在背景窗口当前命令行显示：

OK. Find:

(3) 接着键入要查找的字，如 data，再按回车键，即

OK. Find: data↙

True BASIC 将自动地从指定起点向程序结束方向查找，并将光标移到此后第一次"data"出现的位置上. 如果还要继续查找 data，只要再按F3键和回车键，而不必再键入 data. 注意，data 中字母不分大写与小写.

如果再查找其它字，则再重复(1)～(3)的操作. 如果被查找的不是一个字，而是一个字的一部分，则用字符串表示，如"dat"，也可以按上述三步操作查找. 注意，这对在字符串中，大写与小

写是有区别的，只查找完全相同的字符串.

12.6.2 Change(置换)命令

用该命令可以把程序中从光标到结束的旧字(或数字)用新字(或数字)全部置换. 命令形式如下:

OK.Change 旧字,新字.

OK.Change "旧串","新串"

12.6.3 Try(试探)命令

TRY 命令非常类似于CHANGE 命令. 但它允许对每次字或数字的出现，是否作置换做出回答. 命令形式如下:

OK.Try 旧字,新字

OK.Try "旧串","新串"

当找到第一个旧字(或旧串)时，光标停下闪动，并在命令行显示如下提问:

Change?

若键入字母 n，则这个位置不被改变，若键入字母 y，则改变. 光标将移至下一个出现的位置，如此下去. 若中断试探过程，可键入字母 q，True BASIC 将停止查找，退出 Try 命令.

12.6.4 行模块编辑

行模块，即被标记的若干行. True BASIC 可以标记行模块，然后可以对它作删除、前后移动、左右移动、编辑等.

(1) 标记一个行模块.

[F4]键是标记开关. 移光标到预定若干行一端的行标志上，按[F4]键，则该行的行标志变亮. 再移光标到预定若干行另一端的行标志上，再按[F4]键，这时预定若干行的光标志均变亮. 这就形成了行模块. 如果把光标再移到行模块的任一行标志上，再按[F4]键，这时行模块的行标志由亮变暗，表示标记已被解除. 一行被标记后，也可以看作是一个行模块.

(2) 删除行模块.

把光标移到行模块的任一行标志上，按[Del]键. 这时行模块的程序行在编辑窗口消失了. 而显示在背景窗口，告诉已被删除. 这时，如果发现有误，及时按[F7]键，可把删除的行模块重新显示出来，程序恢复原状. 注意，中间不得作其它操作，否则是不能恢复的.

(3) 复制行模块.

把光标移至欲复制位置前一行的行标志上，再按[F5]键，则把行模块复制到光标所在位置行的后面，原来的行顺序后移.

(4) 前后移动行模块.

把光标移至欲移至位置前一行的行标志上，再按[F6]键，则把行模块移至到光标所在位置行的后面.

(5) 左右移动行模块.

向左移动行模块，按[<]键(即[⇧]+[< ,])

向右移动行模块，按 [>] 键(即 [↑] + [>/.])

按一次移一格，连续按连续移动.

(6) 编辑行模块.

如果想在一段程序上作字符编辑，可以用如下编辑行模块命令，把编辑限制在指定的行模块上.

OK. Edit ↙

显示：Now editing marked region.

（现在编辑标记区域）

这时，True BASIC将"清除"被标记的行模块以外的编辑窗口上的内容. 表示现在仅编辑标记区域. 作完字符编辑后，把光标移至编辑窗口，并按 [F4] 抹掉行模块标记. 再按 [F2] 键，并键入

OK. Edit ↙

显示：

Now editing whole file(现在编辑整个文件)

这时，编辑行模块操作状态结束，恢复正常.

12.7 程序的调试

评价程序的基本准则，第1条就是正确性. 要使程序正确，调试程序是必不可少的. 这里，"程序的调试"指的是通过编译、运行程序，发现错误、修改错误的过程.

True BASIC 解释用户程序的过程是逐条逐句的检查过程. 键入 Run 命令后，系统程序首先对用户程序从前到后，逐条逐句检查是否符合语法规则，对于不符合语法规则的，发现一条，指出一条. 程序员可以对话方式修改排除各个语法错误. 有关编译、运行错误的解释与分析详见附录 A.

运行时如出现输入数据类型不对，或个数过多等错误，系统都会及时提供出错信息，找出错误原因，及时改正.

一个较长的程序在运行中可能出现多种错误，使得程序无法正常结束，或给出错误结果，这时说明程序还存在逻辑性错误. 排除逻辑性错误是比较困难的. 为了便于调试，True BASIC 提供了两个调试命令，即 Break 和 Continue 命令. 利用 Break 命令可以在程序的某行上设置或拆除断点.

Break 命令的一般形式如下：

OK. Break 行序号

OK. Break 程序块名

如果参数是程序块名，则表示程序块第1行. 执行 Break 命令，如果指定行上没有断点标记(▶)，则设置断点标记；如果已有断点标记，则拆除断点标记，即恢复正常. 如果断点设置在某一不可

执行语句上(如注释行或 DIM 语句行),True BASIC 将把断点确定在此行后的第一个可执行的行上.

当程序执行到断点时,程序运行暂停. 这时可以按 F2 ,移光标到背景窗口,再用

OK. **Print** $v_1, v_2, \cdots, v_n$

显示变量 $v_1, \cdots, v_n$ 的当前值,还可用

OK. **Input** $v_1, v_2, \cdots, v_n$

修改变量 $v_1, \cdots, v_n$ 的当前值. 改好后可以用继续命令

OK. **Continue**

拆除当前断点,继续运行程序. 注意,当程序执行了 STOP 或END 语句后,用 Continue 命令是无力恢复运行的.

例如:

```
1  PROGRAM ex12_1
2  ! An erroneous program.
3  !
4  DIM a(5)
5  FOR i=1 to 5
6     LET a(i)=i
7  NEXT i
8  END
```

设置断点:

OK. break 1 ↵　　　　(设置断点在第一行上)

显示:

```
1  ▶ PROGRAM ex12_1
2    ! An erroneous program.
3    !
4    DIM a(5)
5    FOR i=1 to 5
6       LET a(i)=i
7    NEXT i
8    END
```

这时在第一行前出现一个标记▶.

调试运行:

OK. Run ↵

Breakpoint.　　　　(停在断点,这里实际停在第 5 行)

OK. print i ↵　　　　(打印 i)

0

OK. continue ↵　　　　(拆除断点,继续进行到 END 停止)

OK. print i ↵　　　　(打印 i)

6

说明,当断点设置在非执行语句上时,运行时将停止在断点后第一个可执语句上.

一个未编译过的程序,运行后或运行中断后,程序中所使用的变量仍维持当前状态,一般情况,也可用True BASIC的简单语句作为命令获得变量的值,或进行处理.但这时不能用CONTINUE命令继续运行.

12.8 汉字输入与输出简介

True BASIC汉化版已经问世,它是在中文磁盘操作系统(CC-DOS)支持下工作的.汉字输入与输出是把汉字作为字符串处理的,包括键盘键入、存储、传送和打印输出.由于汉化工作还不够完善,通用性较差,下面介绍的操作方法较为通用.如果读者使用设备和下述系统不尽相同,读者可参照有关说明,在自己使用的系统上试试.

本节介绍的汉字输入与输出方法使用的是如下微型机系统:

IBM PC 微型机(内存512 K).

IBM 彩色显示器.

FX-100 打印机.

CC-BIOS 2.10中文操作系统(中国电子工业部第六研究所).

True BASIC汉化版(中国科学院软件所).

12.8.1 汉字的输入

学习汉字输入的关键是要搞清机器当前处于什么状态,以及输入方式的转换方法,如图12-7所示.

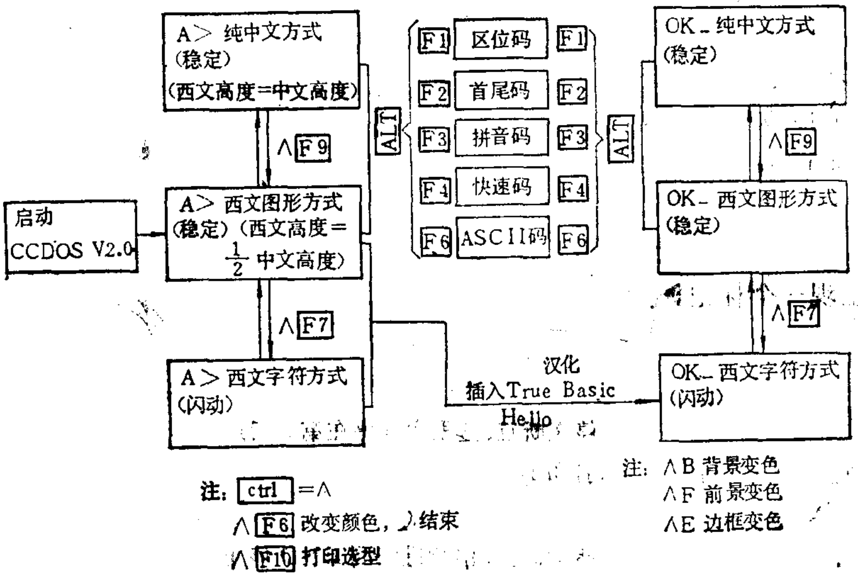


图12-7 输入方式的转换

首先用中文操作系统启动．启动后系统总是处于西文图形方式，可以输入汉字．然后取出中文操作系统盘插入汉化 True BASIC 盘，键入 Hello 命令，进入 True BASIC 状态，系统处于西文字符方式，不可输入中文．如果在处理的文件中不使用汉字，则不必转入西文图形方式．本章前面几节介绍的 True BASIC 系统操作皆可使用，不再重述．

如果在处理的文件中使用汉字，则必须按[Ctrl]-[F7]键转入西文图形方式．注意，这时处理文件的一切操作命令还是和西文(字符)方式完全一样．一切命令要在 OK 提示符或 ASCII 状态下键入．只有在输入汉字时才进入中文输入状态(区位码、首尾码、拼音码或快速码)．这里，我们以汉语拼音码输入方法为例，简要地说明一下汉字输入过程：

(1) 键入[Alt]+[F3]进入汉语拼音码输入方式．

(2) 输入汉语拼音码．

汉语拼音字母用小写英文字母表示．为了提高汉字输入速度，在 CC-BIOS 支持下，一个汉字拼音码最多允许击三次键．为此系统对一些声母和韵母作了简化规定，如表 12-2 所示．

表 12-2

声韵	声母			韵母								
拼音	zh (知)	ch (吃)	sh (诗)	an (安)	ang (昂)	ao (熬)	ai (哀)	en (恩)	eng (亨)	ing (英)	ong (轰)	ü (吁)
对应键	a	i	u	j	h	k	l	f	g	y	s	v

用户应熟记这些键与拼音中声母和韵母的对应关系，以便正确输入．例如，输入一个汉字“欢”，则应键入“huj”，而不应输入“huan”．此外，凡不足三个键的拼音输入可以用字符“[”作结束标志，也称“[”为结束键，以便减少重码．例如，输入一个汉字“迎”，最好键入“yy[”，这比输入“yy”要快．

(3) 汉字选择．

在汉字输入方式下，屏幕最后一行将出现汉字输入提示．格式如下：

××:	000	▌1 ▌2 ▌3 ▌4 ▌5 ▌6 ▌7 ▌8 ▌9 ▌0	[……]
输入方式	汉字编码	重码汉字	重码汉字个数

当输入一个汉字编码后，在提示行上会显示重码的汉字，要在重码汉字中选择所要的汉字．按[>]键，可以显示后边的汉字，10 个字为一页，一页一页的向后(左)翻动，按[<]键，则向前翻动．若越过端点会自动报警．

如在当前行中，首次选择汉字，则可按下对应的数字键；如需在同一页上再选择任意汉字，则要先按[Alt]键，再按下对应数字键；如在当前页已选取了汉字，进而希望在前页或后页中选取汉字，则可以使用恢复前页([Alt]+[-]) 和恢复后页([Alt]+[=]) 操作，以达到目的．

12.8.2 汉字输出

汉字输出和西文字符串输出一样．例如，把汉字作为字符串放在打印语句之中．

```
PROGRAM ex12_2
OPEN #1:printer
PRINT #1:"欢迎指导"
PRINT     "欢迎指导"
END
```

运行结果:

欢迎指导

不仅在屏幕上显示出"欢迎指导",同时可在打印机上打印出"欢迎指导". 另外,屏幕上显示的汉字也可以用硬拷贝(按[↑]-[PrtSn])打印出来.

附录A 错误信息表

一般情况下，True BASIC 处理的错误信息可分为三类：

命令错.

编译态出错.

运行态出错.

当给 True BASIC 不正确的命令时，就会发生命令错. 当 True BASIC 编译源程序时，所出现的错误称为编译态出错. 当程序书写完毕开始运行时，所捕捉到的任何错误均称为运行态出错. 在附录A.1中，对于各种各样的出错，都给出有关出错信息的解释(这里有的出错没有标出出错号).

这个附录包括两个 True BASIC 出错信息表. A.1 按字母顺序排列，A.2 以运行态出错代码为序排列.

A.1 出 错 解 释

Argument types don't match.

当调用带参子程序时，此错可能出现在过早地使用这个子程序，也就是说有关的子程序还没有定义. 或者是使用不同的参数来调用同一个子程序，即：有可能调用的参数个数不对，或类型不一致. 此时应核对一下子程序中参数变元的定义及调用时所使用的相应值.

Bad FIND item; try using quotes.

当试图寻找一包含逗号或双引号的符号串时，被查找的串必须用双引号括住(此规则在 INPUT 语句和 DATA 语句也具有同等效力).

Badly formed USING String. (8201)

在 USING $ 函数或 PRINT USING 语句中所使用的格式串有错. 至少有一项是不附合 True BASIC 规则的. 见 PRINT USING 语句.

Badly formed input line. (8105)

在回答 INPUT 语句时数据格式不正确(它不是用于读磁盘就是用于键盘输入). 最常见的错误是丢失了双引号.

Can't change that key.

True BASIC 不允许改变一般显示字符的键的含义.

Can't Continue.

通常可以用 CONTINUE 命令来继续运行一个暂停的程序. 然而，有时 True BASIC 是拒绝此动作的. 有这样一些可能的原因：当再次运行之前修改了此程序；当程序是由于出错而停止运行的；在使用 CONTINUE 命令之前用过 DO 命令. 这在调试程序时经常碰到. 上述一切均为不能再继续运行此程序了.

Can't copy region itself.

True BASIC 的编辑程序不允许复制自身的某些程序行. 例如，把某个子程序再复制到那个子程序的本身之中. 如果需要这样做，应先把此子程序复制到别处，改名后，再复制到原来的子程序中来，然后再做其它整理.

Can't do graphics on this computer. (11000)

所用计算机不能画图．是由于它没有图形卡，或者没有使用相应的 SET WINDOW．PLOT，DRAW，BOX，FLOOD，GET POINT 或 GET MOUSE 语句．

Can't edit Compiled program.

当程序被编译以后，就不能再被修改了．一旦使用过 COMPILE 命令，就只能用 RUN，SAVE 或 REPLACE 命令来处理这一程序．如果还想改动，应事先留有副本，再编辑以前的副本．

Can't erase file not opend as OUTIN. (7301)

如果文件没有用 ACCESS OUTIN 打开，就不能使用 ERASE 语句．见 9.2 OPEN 语句及其文件存取的描述．

Can't find library name.

在使用 LIBRARY 语句时应有一文件名．但此文件当前盘上不存在．应检查一下记录此文件的磁盘是否已插入计算机上了．

Can't help with that. Try HELP TOPICS.

当求助回答时，不用 HELP 命令就得用 HELP 键，否则 True BASIC 是不能对所提的问题给以帮助的．当用 HELP TOPICS 时，首先应拼写正确(不能缩写)．再之 True BASIC 盘要在机器中．键入后 True BASIC 才会对此问题给出完整的解释．

Can't inpue from OUTPUT file. (7303)

用 ACCESS OUTPUT 打开的文件是不能用来读取数据的．如果需要读，则应把 OPEN 语句选择项改成 ACCESS OUTIN．

Can't invert singular matrix. (3009)

当使用矩阵函数 Inv 时，若对奇异阵施行求逆是不行的．奇异阵在 True BASIC 中规定不能求逆．

Can't output to INPUT file. (7302)

用 ACCESS INPUT 打开的文件，是不能向它写入数据的．如果需要写，则应把 OPEN 语句选择项改成 ACCESS OUTIN．

Can't PRINT to middle of text file. (7350)

正文文件内不能重写数据．要写只能用 RESET 语句把文件指针移到文件的末尾再写，或者在删除此文件后再重写．

Can't SET RECSIZE on non empty record file. (7250)

当某一记录文件被建立以后，它就具有固定的记录长度．在改写或重读此记录文件时，不能改变文件的记录长度．

Can't SET WINDOW in picture. (11004)

图画是不能重新设置窗口或屏幕坐标的．在图画中不能使用 OPEN SCREEN 或 SET WINDOW 语句．

Can't use ANGLE(0,0). (3008)

不能定义 ANGLE(0,0)，其中的参数至少要有一个是非零的．

Can't use this Statement here.

在使用 True BASIC 结构语句时，不能写错位置．例如 SELECT CASE 语句中的 CASE 写错了位置，或丢失了；IF-THEN 语句中用错了 ELSE IF；SELECT CASE 行上多加了不必要的语句，等等．请参见本书中相应语句描述的章节．

Can't use # 0 here. (7002)

对 # 0 通道是不能使用 OPEN 或 CLOSE 语句的，因为 # 0 通道总是打开的．

Channel is already open. (7003)

当试图打开某一通道时，它实际上已经打开了．在程序中应检查一下这一通道的使用情况．所以要记住：

当不使用某一通道时，应及时关闭它.

Channel isn't a window. (11005)

当打开或关闭某个窗口而使用了 WINDOW 语句时，因为所使用的通道已与文件相联，而误认为是某个窗口. 再核对一下 WINDOW 语句中所给定的通道号使用是否正确.

Channel isn't open. (7004)

当用某个通道存取文件或窗口时，相应的通道必须已打开. 在使用通道时，必须用 OPEN 语句打开它. 核对一下所给定的通道号是否正确，以及它是否真的打开了. 还有是否用 CLOSE 语句顺便关上了此通道.

Channel number must be 1 to 1000. (7001)

通道的数值必须选在 1 到 1000 范围内，通道数值不能在此范围之外.

(Compiled program.)

这是一个提示显示. 它出现在给出 CompILE 命令之后. 编译程序调出来时，在编辑窗口上显示出来的. 由于编译程序把源程序代码整理成内部形式，故不能再看到编译前的程序正文，也不要再修改它，否则就应先留有副本再编译或根本别编译了.

Constant too large: constant.

对所用的计算机来说，被显示的数值太大而无法处理. 可用 PRINT MAXNUM 命令显示计算机所能表示的最大数值. 此时应考虑修改有关常数.

Cursor set out of bounds. (8601)

SET CURSOR 语句给出了行和列的位置. 它们表示光标当前应在的窗口范围. 我们记得，每个窗口有它自己的字符坐标. 由于左上角位置总是行为1，列为1. 所以只要表示出右下角即可. 可用 ASK MAX CURSOR 来找出行、列数值.

Data item isn't a number. (8101)

要从记录文件中读取一个数，但此时文件的记录项确是一个字符串. 故不能给数值变量赋值. 可能是定位有问题.

Data item isn't a string. (8104)

要从记录文件中读取一个字符串，但此时文件的记录项确是一个数. 故不能给串变量赋值. 也有可能定位错.

Det needs a square matrix. (6002)

Det 函数仅能用于方阵. 这是受数学定义所限.

Disk full. (9006)

当把文件逐个向磁盘写入时，所占的磁盘空间越来越大，直至不能再装. 这种错一般由操作系统直接给出.

Diskette removed, or wrong diskette. (9005)

若已经打开了某个文件，True BASIC 将要使用它，但却把磁盘取出换上了其它盘，所以出错. 已经正在使用的磁盘不可从驱动器中取出!

Division by zero. (3001)

有一个表达式是打算用零来除某一数值. 如果想用它来代替最大允许值，还不发生错误，那就得使用 WHEN 语句来括住此表达式. 见下例：

```
When error in
    let x=(1+2+3)/0
use
    let x=Maxnum
end Whe
```

Maxnum 是一个 True BASIC 函数，在计算机上它可以给出最大允许值.

Doesn't belong here.

光标在程序中所处的位置不正确，有些语句在使用中会碰到这种情况. 应把光标移到相应菜单的适当语句上. 调整光标到正确位置并继续.

Ending doesn't match beginning.

在所使用的某些结构语句中，例如 FOR-NEXT 或 IF-THEN-ELSE 必须开头与结束相呼应. 通常是丢掉了 END 语句，或者是在循环语句中头、尾的控制变量不相呼应. 应仔细地阅读这些程序行，找出不匹配的地方.

Error in PLAY string. (4501)

在 PLAY 语句中所给定的符号串不符合 True BASIC 文法. 见 10.1 节关于写 PLAY 串的详细描述.

Expected"thing".

当前光标所指向的位置不是 True BASIC 允许的字或标点符号，而是一些其它的东西. 这一信息对于重新修补某语句是有益的.

Expected relational opertor.

当前光标所指的位置，必须放上某关系运算符，例如"="或"<"等等. 最终，程序在此必须写出某种比较运算(注意，True BASIC 不允许如同"IF a THEN…"或其它版本的"IF a < > 0 THEN…"语句格式).

File already exists. (9004)

当建立某一新文件时，实际上它已存在. 应核对一下有关的文件名. 如果是打算重写某个已存在的文件，那么在 OPEN 语句中应把 CREATE NEW 改写成 CREATE NEWOLD. 不能用 SAVE 命令而要用 REPLACE 命令.

File is read or write protected. (9001)

当试图读、写某文件时，文件处于读或写保护状态. True BASIC 不会处理读或写保护文件，所以应退出 True BASIC 解除操作系统对此文件所做的读或写保护动作，或去掉软盘读写保护.

File Pointer out of bounds (7252)

当打算用 RESET 或 SET POINER 或 SET RECORD 语句来改变文件的指针时，但所给定的文件指针不是比 1 小，就是超出了文件结束符. 可用 ASK FILESIZE 语句查出文件有多长.

File's record size doesn't match open OPEN RECSIZE. (7103)

每个记录文件都由它的 RECSIZE 建立相应的记录长度. 当被打开文件的 RECSIZE 与在 OPEN 语句中给定的 RECSIZE 不一致时，就出现此问题. 如何才能正确地打开此文件呢？最简单的方法是以文件的记录长度为准，把 OPEN 语句中的 RECSIZE 部份删掉. 若还想在 OPEN 语句中使用 RECSIZE，那么可以先执行一小段检测程序，查出文件的 RECSIZE，然后，赋值再打开它. 这样就会用到 ASK RECSIZE 语句.

IDN must make a square matrix. (6004)

单位阵(幺阵)必须是方阵. 另外当使用 IDN(x, y)函数时，必须 x=y.

Illegal array bounds for name in routine.

在数组的定义中，上界小于下界是非法的. 核对一下相应 DIM 语句所设置的数组(实际上，True BASIC 允许下界比上界大 1，但比 1 大是不行的，这时所定义的数组是不能含有元素的).

Illegal Chamel.

在使用 SET 或 ASK 语句时，通常应给出所需的通道号. 这里是由于没有给而非法，应该查出加上. 但有时在 SET 或 ASK 语句中又不允许有通道号，可是却加了，当然应查出删掉.

Illegal data.

这是 DATA 语句书写不当. 逗号只能放在两个数据项之间，而不能放在某个数据清单的后面. 所有的双

引号必须成对书写。

Illegal exit.

书写 EXIT DO 语句是为了退出某个 DO 循环；书写 EXIT FOR 语句是为了退出某个 FOR 循环。若不想标明是何种循环时可用 EXIT FRED. 查一下 EXIT 语句在相应结构中是否使用的正确，主要是 EXIT 语句的后缀。

Illegal expression.

光标当前所指的表达式位置有不符合 True BASIC 规则的地方，应检查一下是否用了双操作符（例如，"1++2"），或在一个数值中加入了逗号（例如，"1,000"）。另外，所使用的变量名是否合法，等等。

Illegal Keyword.

在此，光标指出关键字用得不正确的位置。例如，在 BOX 语句中是否丢失了 LINES, AREA, ELLIPSE 或 CLEAR 关键字。检查相应语句，并改正它。

Illegal number.

光标所指位置需要一个数值，但没有给出，还可能所书写的数值不符合 True BASIC 文法。有的时候 True BASIC 要求某数值常数是固定类型，不得用其它类型，而你用了。比如，DIM 和 OPTION BASE 语句中只能用整数常数。

Illegal option.

True BASIC 只提供了两种选择语句 OPTION ANGLE 和 OPTION BASE. 你可能在拼写 ANGLE, BASE, DEGREES 和 RADIANS 时不恰当。

Illegal parameter.

当使用 SUB, DEF 或 PICTURE 语句时，是为了定义一个子程序，有时在其中会用到参数，当使用了同名参数，或与某简单变量名重名时，会出现此错。

Illegal statement.

True BASIC 的每一种语句都必须以关键字开头，象 LET, SELECT 等等. 核对一下所拼写的关键字是否正确。有些 BASIC 版本在书写赋值语句时可以省掉 LET 关键字，但 True BASIC 是不允许省掉的。

Improper ORD string. (4003)

ORD函数所需要的不是一单符号构成的串；就是由 ASCII 码给定的合法名字串，允许不连通或加有空格。

Improper NUM string. (4020)

用于描述 NUM 函数的串不能表示为 IEEE 8-字节的浮点数，检查一下建立的是否正确。

***Input item bigger than RECSIZE.* (8302)**

此条信息指出有故障的记录文件，它说明文件中的某些记录要比文件定义的记录长度长. 如果磁盘上有这种带变异性的文件而且用到它，此信息就会出现，或者没有使用 True BASIC 文件，而是干预了其它不明了的字节文件，也有可能发生。

INV needs a square matirx. (6003)

矩阵变换只限于方阵. 当试图用 INV 函数转换某个非方阵时，是不允许的. 应检查一下矩阵是否是二维，且每维是否都有相同的长度。

LBOUND out of range. (4008)

当用 LBOUND(A,3) 调用时，数组 A 不能是三维以上的. 检查一下相应的维数使用情况。

LOG of number<=0. (3004)

算术对数自变量不能取负值。

margin less than zonewidth. (4006)

SET MARGIN 语句中所设置的 margin 比 zonewidth 小是不允许的. 应检查一下相应的变量值。

Mismatched array sizes. (6001)

在使用 MAT 语句时需要相同大小的数组,但却用了不同长度的数组. 例如,矩阵加法就需要两个数组有相同的长度,因为它要一起相加. 矩阵乘法就要有更复杂的规则,也应有相应等同长度的数组.

Missing end statement.

整个 True BASIC 程序必须包含 END 语句,但当前却丢掉了. 在合适的位置加上它再试一下.

MOD and REMAINDER Can't have 0 as 2nd argument. (3006)

Mod 和 Remainder 不允许零作为它们的第二个参数,因为这是等于用零来除某数. 应仔细检查一下所用的参数.

Must be a function name.

当书写 DEF 或 FUNCTION 语句时,没有在相应关键字后面跟有适当的函数名.

Must be a number.

几乎在任何地方 True BASIC 都允许数值表达式或用简单数值,但有些地方只能用数值. 例如,CASE就就不能用表达式,只能用数值常数. 如果必须用表达式,则应把 SELECT CASE 结构,改成 IF-THEN ELSE 结构.

Must be a Picture name.

DRAW 语句应用名字,而且是图画名字,不能用其它的名字,检查一下程序中名字的使用.

Must be a string constant,

True BASIC 几乎在任何地方都可以使用符号串常数来代替串表达式,但反之例外,例如,CASE 测试就不能用串表达式,如果一定要用,只能把 SELECT CASE 结构改成 IF-THEN-ELSE 结构.

Must be a subroutine name.

CALL 语句仅能用于调子程序,所给的串必须是子程序名. 检查一下所使用的名字是否正确.

Must be a variable.

在应当使用变量的地方,使用了表达式或子程序名. 例如:LET, INPUT 和 GET 语句就一定要用变量. 检查一下这些语句使用的是否正确,是否与子程序、函数、图画重名.

Must be an array.

在 True BASIC 程序中,很多地方都必须使用数组变量,而不是某一简单变量. 例如:MAT 语句是仅使用在数组上,许多函数,如 Lbound 和 Size 也只能在数组上使用,因此应核对一下数组的名字是否拼写正确,是否与 DIM 语句中定义一致.

Must be byte file for READ BYTES. (7351)

在正文文件或记录文件中使用了 READ BYTES 语句,这是不行的;这些文件必须是字节文件,才可能按序读写. 检查一下 OPEN 语句中为打开这一文件所包含的 ORGANIZ ATION BYTE 部份.

Must be byte file or empty for SET RECSIZE. (7251)

SET RECSIZE 仅能用于字节文件或空白的记录文件,正文文件不能有记录长度,非空的记录文件有它自己的记录长度. 检查一下所使用的通道是否正确.

Must be record or byte file for SET RECORD. (7202)

SET RECORD 不能用于正文文件,可以重新设置一正文文件的指针指向文件的最开始或文件的末尾.

Must be record or byte file. (8502)

READ 或 WRITE 语句不能用于正文文件或一个窗口,它们只能使用在记录和字节文件. 核对一下所使用的通道是否有上述情况. 对于正文文件及窗口只能使用 INPUT 和 PRINT 语句.

Must be text file. (8501)

INPUT 或 PRINT 语句不能用于记录文件或字节文件,它们只能使用在正文文件中. 记录文件或字节文

件应使用 READ 或 WRITE 语句. 可用 ASK ORGANIZATION 询问一下文件的类型，再检查是否使用正确.

Must SET RECSIZE before WRITE. (8304)

在向一空记录文件上写入时，没有定义文件的记录长度. 在第一条 WRITE 语句之前必须执行 SET RECSIZE 语句，或用 OPEN 语句指定文件的记录长度.

Name can't be redefined.

对任何两个不同的实体均不可使用同一个名字. 比如，如果变量名定义为 X，那么子程序名就不能是 X，对于重名 True BASIC 总是判出错.

Negative number to non-integral power. (3002)

当计算 n 的x 次幂时，n 为负数且 x 为非整数，其结果在数学上无意义.

No CASE Selected, but no CASE ELSE. (10004)

在执行 SELECT CASE 语句时，CASE 测试不成功，由于没有 CASE ELSE 语句块来捕捉此问题，True BASIC 就显示上述信息. 检查一下所选择的常数是否使用得齐全. 若加上 CASE ELSE，则在测试时可处理各种情况的问题.

No GET MOUSE on this computer. (8700)

在程序中使用了 GET MOUSE 语句，但在计算机上并没有加上一鼠标定位装置.

No main Program.

在当前的程序文件中，仅有函数、图画和子程序，但没有主程序. True BASIC 无法执行这样的程序.

No saved copy-still want to compile?

当给过 COMPILE 命令后，True BASIC 将冲掉当前 ASCII 码文件，而以不可见的编译代码代之. 为了提醒注意，事先做好必要的复制工作，特出示此信息. "n"为退出此命令；"y"为真正开始编译. 故在回答时，应小心地检查一下是否留下了 ASCII 码文件副本.

No such color. (11008)

用 SET COLOR 语句所设置的颜色名 True BASIC 不认识. 颜色名可用大写，也可用小写，但不能在名字中使用多余的符号. 再用 True BASIC 能认识的颜色名完整清单，检查一次程序.

No such file. (9003)

要打开的文件实际上不存在，从很多命令(例如 OLD)中也可能得到这个信息. 首先应检查一下名字是否用对了，再检查有关的磁盘是否插入到驱动器里了. 可用 FILES 命令配合检查一下当前的磁盘内容.

(True BASIC 也象一些操作系统一样使用了扩展名，例如，所有的 True BASIC 程序都自动地加上了. "TRU"的扩展名，这一点也应引起注意.)

No such function or subroutine.

在某命令中给出了某个函数、子程序或图画的名字，但这些子程序却不存在. 例如可键入 LIST myfunc，而名字也确实拼写正确了，但 myfunc 不存在.

No such line numbers.

命令中所给出的行号与实际程序中的行号范围不符.

No USING item for output. (8202)

在使用 Using $ 函数或 PRINT USING 语句时，应有格式描述串，但是这里没有，而仅有有关的正文.

Not found.

可以使用 FIND 键来查找某些字或短语，但并没有找到. 记住，True BASIC 总是匹配整字. 如果不打算找一整字，象 'print'，而是要找 'i'，那么必须用双引号括住所要查找的有关内容. 例如:

Find: "i"

这样，True BASIC 将在'print'中找出'i'来．还要记住，True BASIC总是从当前光标位置开始一直找到文件的末尾．如果所要找的内容在光标的前面，那应先移动光标，再作查找．

ON index out of range, no ELSE given. (1000)

当用了 ON GOTO 或 ON GOSUB 语句时，所指定的分支号出界了．例如，在用"ON x GOSUB 100,120,130"时，若 x 比 1 小或比 3 大，就会出现这样的信息．应核对一下相应控制变量的值．

Out of memory. (5000)

这时问题出在要求提供的存储器比当前机器上所支持的要多．这时可做一些在小存储器环境下的实验．例如：

采用小数组．多维数的数组可化简成一维数组分块处理．

编译源程序．因为 True BASIC 编译代码远比源程序代码短得多．

检查一下所有的子程序，能采用递归调用的尽量写成完美而合法的递归子程序．这也是常用且有效的方法．

还可以试用一下 Packb 和 Unpackb 函数作为最后的手段，它可以压缩数据而少占内存．

Output item bigger than RECSIZE. (8301)

当向一记录文件中写某数据项时，由于文件建立时设置了固定的记录长度，因而不能写入超长的数据．可用 ASK RECSIZE 找出文件的记录长度．记住，任何数据项均不能超过记录长度的指定值．每个数值只能是 8 字节长．

Overflow. (1000)

当要计算的数值超出计算机所能处理的范围时就溢出．用 PRINT MAXNUM 查计算机所能使用的最大数．如果打算通过溢出来获得尽可能大的数值，而又不出错，可写成：

```
when error in
    let x=Tan(pi/2)
Use
    let x=Maxnum
end when
```

Please say what lines you mean.

假如键入了 MOVE 命令，但又没有描述所要移动的行位置，可用 MARK 键来标记行，或者带上移动的行位置说明再键入 MOVE 命令．

Please try"CHANGE old, new".

当修改了一短语时，必须同时给出原来的短语和要改的短语．在任何短语中若包含有逗号或双引号时，都必须用双引号括住其整个内容．

Please try"DO filename".

当用 DO 命令时，必须给出文件名．重新键入带文件名的 DO 命令！

Please try"INCLUDE filename".

当用 INCLUDE 命令时，必须给出文件名．重新键入带有文件名的 INCLUDE 命令！

Please try "KEY" or "KEY FROM file" or "KEY TO file".

KEY 命令有三种形式．关键字本身是恢复原来键的含义，"KEY TO"和"KEY FROM"可更改键的定义．详见参考手册的 16.4 节．

Please try "LOCATE item".

当想确定某短语时，必须给出这个相应的短语．

Please try "OLD filename".

当使用 OLD 命令时，必须给出所要装入的文件名，以便 True BASIC 从磁盘中调出此文件．

Please try "SAVE filename" or "REPLACE filename".

当对当前文件记盘时,必须给它一个名字,以便在磁盘目录中登记.

Please try "UNSAVE filename".

当删除某文件时,必须给出这个文件名,以便在磁盘目录中去掉此文件.

Please type line numbers as 100 or 100—150.

假如所给出的命令是 DELETE, 则所带的参数应是一个行号或由行号表示的行模块. 不带后面这些说明项 True BASIC 就不懂. DELETE 100 是删除第 100 行. DELETE 100—150 是删除 100 到 150 行的程序块.

Program stopped.

按 STOP 键, 即 Ctrl - Brear. 程序就被终止运行,不按此键程序将继续工作.没有停止运行的其它方法.

REPEAT $ count<0. (4010)

在使用 Repeat$(a$,n)函数时,n 不能是小于零的数. 检查一下所使用变量的赋值情况.

Reading past end of data. (8001)

在执行 READ 语句时,所读的 DATA 项比定义给出的 DATA 项要多,造成读出界. 解决的办法有二种. 其一是使读的项与定义的项在个数上一致;其二是读之前可用 MORE DATA 测试一下,再决定是否继续读.

Reading past end of file. (8011)

当从文件中读多于文件所记录的内容时出此错,适合于各种文件. 防止的方式有二. 其一是明确文件的尺寸及记录格式;其二是可用 MORE #n 和 END #n 预先测定相应文件是否结束了.

RETURN without GOSUB. (10002)

当执行 RETURN 语句时,由于没有对应的 GOSUB 语句事先执行过, 所以出此错. 追踪此错有时是很困难的,建议使用子程序名字调用方法,这样的结构清晰,而且不容易发生类似错误.

Screen bounds must be 0 to 1. (11003)

以 OPEN SCREEN 语句设置的屏幕边界必须在 0 到 1 的范围之内. 不管屏幕有多大, 其左下角定义为 0;右上角定义为 1.

Screen minimum>=maximum. (11002)

OPEN SCREEN 语句定义了新窗口, 用四个数字表示,它们顺序表示为: 左、右、下和上. 右边界必须大于左边界; 上边界必须大于下边界. 如果打算获得可变换的窗口, 就不能使用此语句, 而应使用 SET WINDOW 语句.

Size index out of range. (4004)

假如使用的是函数 Size(A,3), 当数组 A 的维数小于 3 时就会出现此错. 应检查 DIM 语句定义的数组维数是否正确.

Sqr of negative number. (3005)

不能对负数求平方根!

Statement outside of program.

光标当前所指的程序行既不属于主程序,也不属于外部任何子程序、函数或图画, 是多余的程序行. True BASIC 不允许这样的程序行出现. 主程序一定要以 END 语句结尾. 比如,有一程序在 END 语句之后, 且又没有说明它是子程序或外部子程序,则出错.

String given instead of number. (8103)

企图用 INPUT语句输入某个数值,在此又没给出数值,而给出了符号串,属于类型错.如果是从文件中读数据,应检查一下文件内数据类型与此语句中变量类型是否一致. 如果是从键盘上读数据, 且读的是符号串而不是欲读的数值,可用 Val 函数将它转换成数值.

String too long. (1051)

所建立的串长度大于 32000 个字节.

Subscript out of bounds. (2001)

向数组中赋值时,超出了数组边界. 可用 Lbound 和 Ubound 查出数组的相应边界,且在屏幕上显示出来.

TAB column not between 1 and margin. (4005)

在 PRINT 语句中使用 Tab 函数时,所给定的参数不是 1 至当前范围之内,出此错. 可用 ASK MARGIN 语句找出当前的范围.

The BYE command is just "BYE".

当退出 True BASIC 时,只能键入"BYE",而不能带有其它任何东西.

The COMPILE command is just "COMPILE".

使用 COMPILE 命令时,不能附带任何其它东西.

The CONTINUE command is just "CONTINUE."

使用CONTINUE命令时,不能在此命令上附带任何其它东西.

The RUN command is just "RUN".

运行当前的程序时,只能键入"RUN",不能附加任何其它东西.

This must first appear in a DIM or DECLARE DEF.

对于任何数组和外部函数来说,必须先定义后使用!

Too few input items. (8002)

在执行 INPUT 语句时,输入所包含的项数比在 INPUT 语句中要求的项数少,如果输入来自键盘,将要求继续输入,如果来自文件,就应核对一下文件中的项数与语句中的项数不一致的原因,然后做出相应的改动.

Too many channels open. (7102)

如果想打开十个以上的通道是不行的. True BASIC 在同一时刻只能处理十个通道,对于一些暂时不用的通道应即时关闭(某些操作系统仅可以同时打开小于十个的通道,可查阅一下用户指南,看看你所用的计算机能打开多少).

Too many input items. (8003)

在执行 INPUT 语句时,输入所包含的项数比在 INPUT 语句中要求的项数多,如果输入来自键盘,则忽略所有多余项,如果来自文件,就应核对一下文件中的项数与语句中的项数不一致的原因,然后做必要的改动.

Trouble using disk or printer. (9002)

True BASIC 在某个磁盘上或打印机上发现问题. 不同的计算机其出错的原因也不尽相同. 首先应检查下电源是否接通,无论是打印机还是磁盘驱动器都是如此 另外,打印机的纸是否装好,它是否处于暂停态;驱动器中是否没插入磁盘或磁盘是否插反了等等. 如果始终找不到错,应在计算机上运行诊断故障程序帮助解决.

Type is wrong for name in routine.

当调用某个子程序时,用的是另一个子程序的名字, 这时会出现相应参数类型不匹配错误. 否则就是把子程序中定义的参数,在调用时把他们的类型搞错了,数值参数变量不能赋字符型,反之亦然. 子程序不能作为函数使用,反之亦然.

UBOUND out of range. (4009)

假如使用了 Ubound(A,3)且 A 是一个小于 3 维的数组就出错. 检查一下 DIM 语句定义,再核对一下所使用的维数.

Undefined routine name in routine.

不能在某个子程序中再定义子程序、函数或图画,也就是说,子程序、函数或图画不能嵌套定义,而只能嵌套使用.

Unknown OPEN option. (7101)

在 OPEN 语句后面的选择部分，没有给出，ACCESS，CREATE 或 ORGANIZATION，这种错误的出现有二：其一是根本没有写上；其二是可能拼写错或用缩写词.

VAL String isn't a proper number. (4001)

所用的 Val 函数中，给定的串必须表示一适当的数.

What? (Please type HELP)

所键入的命令 True BASIC 不懂. 如果想求助于 True BASIC 来解释有关的使用，应键入 HELP，然后回车，就会给出简要的注释. 此时带有 HELP 的 True BASIC 盘应插在磁盘驱动器上.

Window minimum＝maximum. (11001)

当使用某个 SET WINDOW 语句时，所设置的垂直或水平方向，最大值与/最小值相等，True BASIC 不允许这样. 因为在这样的窗口里无法显示任何东西. 也可能问题出在设置边界的顺序搞错了，应该是左、右、下、上.

Wrong number of arguments.

在调用某个子程序、函数或图画时，所给出的调用参数值有错. 检查一下子程序相应的定义及调用的方式.

Wrong number of dimensions.

在使用数组时，维数弄错了！核对一下 DIM 语句定义及使用的位置上的维数是否一致. 如果是把一个数组传给某一个子程序，就要检查一下子程序的参数. 记住，两维数组在子程序参数表中必须表示成 A(,)；三维应为 A(, ,)，等等.

Wrong type.

所定义的任何量必须与使用的相应的量在类型上保持一致. 如，串变量不能赋数值，反之亦然. 记住，字符串进行连接时，应书写“&”而不能使用“＋”.

Wrong type of file. (7104)

True BASIC 所建立的各种文件均有它自己的类型标志，以前记录在磁盘上的文件应与用 OPEN 语句要打开的文件的描述在类型上保持一致. 虽然OPEN 语句使用的合法，但由于与打开的文件类型不匹配也是枉然.

You have two routines called name in routine.

子程序的重名定义为非法. 对于任何不同的子程序必须给以不同的名字，否则 True BASIC 不知道调用那一个. 这种人为的二义性是容易避免的. 核对一下子程序名！

Zero to negative power. (3003)

当求 0 的 n 次幂时，n 不能小于零，这在数学上称无意义. True BASIC 也会给出出错信息.

ZONEWIDTH out of range. (4007)

当执行 SET ZONEWIDTH 语句时，所设置的 zonewidth 比 1 小或比 margin 大. 可用 ASK MARGIN 语句找出当前的 margin.

A.2 出错代码及出错信息

下面是除键入错以外的所有运行态出错信息清单，它们是由 Extype 函数送回的出错代码. 你可以从表中获得出错的简要解释. 所有的出错号均与美国国家标准 BASIC 定义的一致(编者注：此表给出英文注释，以方便与屏幕上出现的内容相对照)，中文解释可从 A.1 中查到.

代码	解释
1000	Overflow.

1051 String too long.

2001 Subscript out of bounds.

3001 Division by zero.

3002 Negative number to non-integral Power.

3003 Zero to negative Power.

3004 LOG of number <=0.

3005 SQR of negative number.

3006 MOD and REMAINDER can't have 0 as 2nd argument.

3008 Can't use ANGLE (0,0).

3009 Can't invert singular matrix.

4001 VAL String isn't a proper number.

4003 Improper ORD string.

4004 SIZE index out of range.

4005 TAB column not between 1 and margin.

4006 Margin less than zonewidth.

4007 ZONEWIDTH out of range.

4008 LBOUND out of range.

4009 UBOUND out of range.

4010 REPEAT $ count <0.

4020 Improper NUM string.

4501 Error in PLAY string.

5000 Out of memory.

6001 Mismatched array sizes.

6002 DET needs a square matrix.

6003 INV needs a square matrix.

6004 IDN must make a square matrix.

7001 Channel number must be 1 to 1000.

7002 Can't use # 0 here.

7003 Channel is already open.

7004 Channel isn't open.

7101 Unknown OPEN option.

7102 Too many channels open.

7103 File's record size doesn't match OPEN RECSIZE.

7104 Wrong type of file.

7202 Must be record or byte file for SET RECORD.

7250 Can't SET RECSIZE on non-empty record file.

7251 Must be byte file or empty for ISET RECSIZE.

7252 File pointer out of bounds.

7301 Cant' erase file not opened as OUTIN.

7302 Can't oup·· to INPUT file.

7303 Can't input from OUTPUT file.

7350 Can't PRINT to middle of text file.
7351 Must be byte file for READ BYTES.
8001 Reading past end of data.
8002 Too few input items.
8003 Too many input items.
8011 Reading past end of file.
8101 Data item isn't a number.
8103 String given instead of number.
8104 Data item isn't a string.
8105 Badly formed input line.
8201 Badly formed USING string.
8202 No USING item for output.
8301 Output item bigger than RECSIZE.
8302 Input item bigger than RECSIZE.
8304 Must SET RECSIZE before WRITE.
8501 Must be text file.
8502 Must be record or byte file.
8601 Cursor set out of bounds.
8700 No GET MOUSE on this computer.
9001 File is read or write protected.
9002 Trouble using disk or printer.
9003 No such file.
9004 File already exists.
9005 Diskette removed, or wrong diskette.
9006 Disk full.
9666 Program stopped.
10001 ON index out of range, no ELSE given.
10002 RETURN without GOSUB.
10004 No CASE Selected, but no CASE ELSE.
10005 *
11000 Can't do graphics on this computer.
11001 Window minimum=maximum.
11002 Screen minimum >=maximum.
11003 Screen boundsmust be 0 to 1.
11004 Can't SET WINDOW in picture.
11005 Channel isn't a window.
11008 No such color.

---

* 此表中的10005号出错是针对CHAIN语句的，它描述的是一种特殊的错误，类似编译态的"Illegal statement"错，但它出现在程序中CHAIN及RETURN位置上。

附录B　ASCII码符号集

附此表是为了便于读者查阅．此表给出的“名称”可作表3-2标准函数Ord的参数．这些ASCII码可在所有True BASIC能运行的计算机上使用．

ASCII 码表

十进制	名称	十六进制	十进进	名称	十六进制
000	nul	00	001	soh	01
002	stx	02	003	etx	03
004	eot	04	005	enq	05
006	ack	06	007	bel	07
008	bs	08	009	ht	09
010	lf	0A	011	vt	0B
012	ff	0C	013	cr	0D
014	so	0E	015	si	0F
016	dle	10	017	dc1	11
018	dc2	12	019	dc3	13
020	dc4	14	021	nak	15
022	syn	16	023	etb	17
024	can	18	025	em	19
026	sub	1A	027	esc	1B
028	fs	1C	029	gs	1D
030	rs	1E	031	us	1F
032	SP	20	033	!	21
034	"	22	035	#	23
036	$	24	037	%	25
038	&	26	039	'	27
040	(	28	041	)	29
042	*	2A	043	+	2B
044	,	2C	045	-	2D
046	.	2E	047	/	2F
048	0	30	049	1	31
050	2	32	051	3	33
052	4	34	053	5	35
054	6	36	055	7	37
056	8	38	057	9	39
058	:	3A	059	;	3B
060	<	3C	061	=	3D
062	>	3E	063	?	3F
064	a	40	065	A	41
066	B	42	067	C	43
068	D	44	069	E	45
070	F	46	071	G	47
072	H	48	073	I	49
074	J	4A	075	K	4B
076	L	4C	077	M	4D
078	N	4E	079	O	4F
080	P	50	081	Q	51
082	R	52	083	S	53
084	T	54	085	U	55
086	V	56	087	W	57
088	X	58	089	Y	59

续表

十进制	名称	十六进制	十进制	名称	十六进制
090	Z	5A	091	[	5B
092	¥	5C	093	]	5D
094	∧	5E	095	—	5F
096	‹	60	097	a	61
098	b	62	099	c	63
100	d	64	101	e	65
102	f	66	103	g	67
104	h	68	105	i	69
106	j	6A	107	k	6B
108	l	6C	109	m	6D
110	n	6E	111	o	6F
112	p	70	113	q	71
114	r	72	115	s	73
116	t	74	117	u	75
118	v	76	119	w	77
120	x	78	121	y	79
122	z	7A	123	{	7B
124	|	7C	125	}	7D
126	～	7E			

附录C True BASIC语句一览表

简单语句　　章·节

ASK MARGIN var……3.2
ASK ZONEWIDTH var……3.2
SET MARGIN expr……3.2
SET ZONEWIDTH expr……3.2
INPUT var1, var2,……3.5
INPUT PROMPT expr$: var1, var2,……3.5
LINE INPUT var1$, var2$,……3.5
LINE INPUT PROMPT expr$: var1$, var2$,……3.5
LET var=expr……2.2
LET var1, var2,…=expr……2.2
LET var$[i, j]=expr$……3.1
OPTION ANGLE DEGREES……3.1
OPTION ANGLE RADIANS……3.1
PAUSE expr……3.1
PLAY music$……10.1
PRINT expr1, expr2,……3.2
PRINT expr1; expr2;……3.2
PRINT USING format$: expr1, expr2,……3.3
READ var1, var2,……5.4
DATA item1, item2,……5.4
RESTORE……5.4
STOP……9.3
SOUND freq, time……10.2

选择结构语句　　章·节

IF condition THEN statement……4.2
IF condition THEN
…
… } block……4.2
…

END IF

IF condition THEN statement1 ELSE statement2 .... 4.2

IF condition THEN

```
    ...  }
    ...  }  block 1
    ...  }
```

ELSE 4.2

```
    ...  }
    ...  }  block 2
    ...  }
```

END IF

```
IF condition1 THEN
    ...  }
    ...  }  block 1
    ...  }
ELSEIF condition2 THEN
    ...  }
    ...  }  block 2
    ...  }
ELSEIF condition3 THEN
    ...  }
    ...  }  block 3
    ...  }
ELSE
    ...  }
    ...  }  block E
    ...  }
ENDIF
```

.... 4.2

```
SELECT CASE expr
CASE test1, test2, ...
    ...  }
    ...  }  block 1
    ...  }
CASE test3, test4, ...
    ...  }
    ...  }  block 2
    ...  }
CASE ELSE
    ...  }
    ...  }  block E
    ...  }
END SELECT
```

.... 4.3 ③

GOTO Line .... 兼 容

IF condition THEN Line……兼容
GOSUB Line……兼容
RETURN……兼容
ON index GOTO Line1, Line2……兼容
ON index GOSUB Line1, Line2,……兼容
ON index GOTO Line1, Line2, …ELSE Line……兼容
ON index GOSUB Line1, Line2, …ELSE Line……兼容

循环结构语句

```
DO
      …
      …  } block……………………………………5.1
      …
LOOP
DO WHILE condition
      …
      …  } block……………………………………5.2
      …
LOOP
DO
      …
      …  } block……………………………………5.2
      …
LOOP UNTIL condition
DO WHILE condition
      …
      …  } block……………………………………5.2
      …
LOOP UNTIL condition
FOR var=first TO Last STEP size
      …
      …  } block……………………………………5.3
      …
NEXT var
EXIT DO……………………………………………5.2
EXIT FOR…………………………………………5.3
```

矩阵语句

章·节

MAT array1=array2 ……………………………………… 6.2
MAT array1,array2,…=array ……………………………… 6.2
MAT array1,array2, …=axpr ……………………………… 6.2
MAT array1$,array2$, …=expr …………………………… 6.2
MAT array1=array2+array3 ……………………………… 6.2
MAT array1=array2−array3 ……………………………… 6.2
MAT array1=array2*array3 ……………………………… 6.2
MAT array1=constant*array2 …………………………… 6.2
MAT INPUT array (?) ……………………………………… 6.4
MAT INPUT array1, array2, ……………………………… 6.3,6.4
MAT LINE INPUT array1$, array2$, ……………………… 6.3,6.4
MAT INPUT PROMPT expr$:array1, array2, ……………… 6.3
MAT LINE INPUT PROMPT expr$:array1$, array2$, ……… 6.3
MAT PRINT array1,array2, ……………………………… 6.3
MAT PRINT array1;array2; ……………………………… 6.3
MAT PRINT USING format$: array1,array2, …………… 6.3
MAT READ array1,array2, ……………………………… 6.3,6.4

程序模块语句

章·节

PROGRAM name ……………………………………………… 2.4
END ………………………………………………………… 2.4
PROGRAM name (var$) …………………………………… 11.3
CHAIN expr$ ……………………………………………… 11.3
CHAIN expr$, WITH (ars$) ……………………………… 11.3
CHAIN expr$, RETURN …………………………………… 11.3
CHAIN expr$, WITH(ars$), RETURN …………………… 11.3
SUB subr (var1,var2,…)

…
… } block ……………………………………………… 8.2
…

END SUB
EXIT SUB …………………………………………………… 8.2
CALL subr (expr1,expr2, …) …………………………… 8.2
DECLARE DEF func1,func2, ……………………………… 8.3

DEF func (var1, var2, …) = expr ……… 8.3

DEF func (var1, var2, …)

…

LET func = expr } block ……… 8.3

…

END DEF

EXIT DEF ……… 8.3

PICTURE pic (var1, var2, …)

…

… } block ……… 8.7

…

END PICTURE

EXIT PICTURE ……… 8.7

DRAW pic ……… 8.7

DRAW pic WITH trans ……… 8.7

DRAW pic WITH trans1*trans2 ……… 8.7

EXPERNAL ……… 8.9

LIBRARY "filename1", "filename2", ……… 8.9

画图语句 章·节

ASK BACK var ……… 7.3

ASK BACK var$ ……… 7.3

ASK BACKGROUND COLOR var ……… 7.3

ASK BACKGROUND COLOR var$ ……… 7.3

ASK COLOR var ……… 7.3

ASK COLOR var$ ……… 7.3

ASK CURSOR x, y ……… 7.6

ASK CURSOR var$ ……… 7.6

ASK MAX CURSOR x, y ……… 7.6

ASK MAX COLOR var ……… 7.3

SET BACK expr ……… 7.3

SET BACK expr$ ……… 7.3

SET BACKGROUND COLOR expr ……… 7.3

SET BACKGROUND COLOR expr$ ……… 7.3

SET COLOR expr ……… 7.3

SET COLOR expr$ ……… 7.3

SET CURSOR expr$ ……… 7.6

SET CURSOR x, y ……… 7.6

ASK WINDOW left right, bottom, top ……… 7.1

SET WINDOW left, right, bottom, top ········ 7.1
OPEN #expr: SCREEN left, right, bottom, top ········ 8.8
ASK SCREEN left, right, bottom, top ········ 8.8
WINDOW #expr ········ 8.8
CLEAR ········ 7.2
BOX CIRCLE left, right, bottom, top ········ 7.4
BOX ELLIPSE left, right, bottom, top ········ 7.4
BOX LINES left, right, bottom, top ········ 7.4
BOX AREA left, right, bottom, top ········ 7.4
BOX CLEAR left, right, bottom, top ········ 7.4
BOX KEEP left, right, bottom, top IN var$ ········ 7.4
BOX SHOW var$ AT left, bottom ········ 7.4
BOX SHOW var$ AT left, bottom USING constant$ ········ 7.4
BOX SHOW var$ AT left, bottom USING expr ········ 7.4
FLOOD x, y ········ 7.4
GET KEY var ········ 7.5
GET MOUSE:x, y, state ········ 7.5
GET POINT:x, y ········ 7.5
PLOT x, y ········ 7.2
PLOT x, y; ········ 7.2
PLOT x1, y1; x2, y2; ········ 7.2
PLOT LINES:x1, y1; x2, y2; ········ 7.2
PLOT POINTS:x1, y1 ········ 7.2
PLOT AREA:x0, y0; x1, y1; …; xn, yn ········ 7.2
PLOT TEXT, AT x, y:expr$ ········ 7.2
MAT PLOT AREA:array ········ 7.2
MAT PLOT LINES:array ········ 7.2
MAT PLOT POINTS:array ········ 7.2

文件处理语句 章·节

ASK #expr:NAME var$ ········ 9.2
ASK #expr:ACCESS var$ ········ 9.2
ASK #expr:RECSIZE var ········ 9.4
ASK #expr:ORGANIZATION var$ ········ 9.2
ASK #expr:FILESIZE var ········ 9.1
ASK #expr:RECORD var ········ 9.1
ASK #expr:POINTER var$ ········ 9.1
ASK #expr:ZONEWIDTH var ········ 9.3
ASK #expr:MARGIN var ········ 9.3

RESET #expr:BEGIN……9.1
RESET #expr:END……9.1
RESET #expr:NEXT……9.1
RESET #expr:SAME……9.1
SET #expr:POINTER BEGIN……9.1
SET #expr:POINTER END……9.1
SET #expr:POINTER NEXT……9.1
SET #expr:POINTER SAME……9.1
SET #expr:POINTER expr……9.1
SET #expr:RECODE expr……9.1
RESET #expr:RECORD expr……9.1
SET #expr:RECSIZE expr……9.4
SET # expr:MARGIN expr……9.3
SET #expr:ZONEWIDTH expr……9.3
CLOSE #expr……8.8, 9.2
ERASE#expr……9.2
OPEN #expr:NAME expr$, ACCESS accmode$
,CREATE crtmode$
,ORGANIZATION orgmode$
,RECSIZE recsize……9.2
OPEN #expr:PRINTER……3.4
INPUT #expr:var1, var2,……9.3
LINE INPUT #expr:var1$, var2$,……9.3
MAT INPUT #expr:arry1, array2,……9.3
MAT LINE INPUT #expr:array1$, array2$,……9.3
PRINT #expr:expr1, expr2,……9.3
PRINT #expr:expr1; expr2;……9.3
PRINT #expr, USING format$: exprz, expr2,……9.3
MAT PRINT #expr, USING format$: array1, array2,……9.3
READ #expr:var1, var2,……9.3
WRITE #expr:var1, var2,……9.4
MAT READ #expr:array1, array2, ,……9.4
MAT WRITE #expr:array1, array2, ,……9.4
READ #expr, BYTES expr:var1$, var2$,……9.5
READ #expr:var1$, var2$,……9.5
WRITE #expr:var1$, var2$,……9.5

出错处理语句 章·节

CAUSE ERROR expr (ERROR=EXCEPTION)……11.1

CAUSE ERROR expr, expr$ ······ 11.
EXIT HANDLER ······ 11.1
WHEN ERROR IN　　(ERROR=EXCEPTION)

```
    ...  }
    ...  }  protected block
    ...  }
```

USE ······ 11.1

```
    ...  }
    ...  }  error handler
    ...  }
```

END WHEN

注：① 用小写字母表示的参数或注释的含义，请见有关章节。

② 兼容——这些语句在原有BASIC各种版本中使用，True BASIC允许使用。由于这些语句在True BASIC中属于被遗弃的，所以，最好不用。对初学者本书未予介绍它们的使用方法。

Document generated by Anna’s Archive around 2023-2024 as part of the DuXiu collection (https://annas-blog.org/duxiu-exclusive.html).

Images have been losslessly embedded. Information about the original file can be found in PDF attachments. Some stats (more in the PDF attachments):
{
  "filename": "XzEwMjgwNTU3LnppcA==",
  "filename_decoded": "_10280557.zip",
  "filesize": 15927825,
  "md5": "d0e5670dc39f834869cea0f0b0c682fd",
  "header_md5": "0ee0cd97d01580d2e5e526c4800fea5b",
  "sha1": "4d5bdc0a178a9d8273e094e8f2669b664bce7944",
  "sha256": "688d4d4ef05d53e360df16dd6ca4f983ed14fdadedbb4657e57a29bafd71569e",
  "crc32": 1051197658,
  "zip_password": "wcpfxk&*^TDwcpfxk",
  "uncompressed_size": 16528913,
  "pdg_dir_name": "",
  "pdg_main_pages_found": 233,
  "pdg_main_pages_max": 233,
  "total_pages": 241,
  "total_pixels": 1689792332,
  "pdf_generation_missing_pages": false
}